# PROCESSING OF THE MANUAL MORSE SIGNAL USING OPTIMAL LINEAR FILTERING, SMOOTHING AND DECODING

Edison Lee Bell

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California


# THESIS

PROCESSING OF THE MANUAL MORSE SIGNAL
USING OPTIMAL LINEAR FILTERING,
SMOOTHING AND DECODING

by

Edison Lee Bell

September 1975

Thesis Advisor: S. Jauregui

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle): Processing of the Manual Morse Signal Using Optimal Linear Filtering, Smoothing and Decoding | 5. TYPE OF REPORT & PERIOD COVERED: Engineer's Thesis; September 1975 |
| | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s): Edison Lee Bell | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940 | 12. REPORT DATE: September 1975 |
| | 13. NUMBER OF PAGES: 160 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | 15. SECURITY CLASS. (of this report): Unclassified |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Morse Code
Kalman Filter
Linear Smoothing
Viterbi Decoder

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis investigates the problem of automatic transcription of the morse signal, and describes and documents several approaches to filtering, processing, and decoding it for transcription. The baseband signal is first modeled as a modified random telegraph wave. A discrete Kalman filter and a linear smoother are then used to process the demodulated signal in order to gain a measure of the effectiveness and

(20. ABSTRACT Continued)

applicability of this model. It is shown experimentally that this model and processing yield a significant reduction in the transcription error rate. Next, a Viterbi decoder algorithm based on a simple Markov model of the code is programmed and tested. Finally, the baseband signal model is incorporated in a more general model for pre-detection Kalman filtering. It is shown that this filter permits acceptable recovery of morse signals whose average signal-to-noise ratio is as low as -14 dB in a 2 kHz bandwidth.

# Processing of the Manual Morse Signal Using Optimal Linear Filtering, Smoothing and Decoding

by

Edison Lee Bell
Lieutenant, United States Navy
B.E.E., Georgia Institute of Technology, 1969
M.S.E.E., Naval Postgraduate School, 1974

Submitted in partial fulfillment of the
requirements for the degree of

ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
September 1975

## ABSTRACT

This thesis investigates the problem of automatic transcription of the morse signal, and describes and documents several approaches to filtering, processing, and decoding it for transcription. The baseband signal is first modeled as a modified random telegraph wave. A discrete Kalman filter and a linear smoother are then used to process the demodulated signal in order to gain a measure of the effectiveness and applicability of this model. It is shown experimentally that this model and processing yield a significant reduction in the transcription error rate. Next, a Viterbi decoder algorithm based on a simple Markov model of the code is programmed and tested. Finally, the baseband signal model is incorporated in a more general model for pre-detection Kalman filtering. It is shown that this filter permits acceptable recovery of morse signals whose average signal-to-noise ratio is as low as -14 dB in a 2 kHz bandwidth.

TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

ACKNOWLEDGEMENTS

I wish to express my deep appreciation to Dr. Stephen Jauregui for his continual support and patience during the preparation of this thesis. I am also grateful to LT. Bill Hickey for his aid in obtaining the performance evaluation of the Pickering 230-D decoder and his comments on operator performance data. Finally I owe special thanks to Mr. Al Wong, Mr. Bob Limes and Mr. Bill Thomas of the Naval Postgraduate School Computer Science Laboratory staff for their valuable assistance in programming and equipment operation.

# I. INTRODUCTION

Economic inflation and the national commitment to the all-volunteer Armed Forces concept have combined to produce unprecedented increases in costs of both manpower and weapon systems in recent years. The public's keen awareness of these higher costs, together with less than enthusiastic support of defense programs in general, has caused Congress to be reluctant to authorize increases in defense expenditures. Thus it has become necessary to reduce the number of armed forces personnel in order to keep defense expenditures within authorized limits. This reduction has had the effect of intensifying the development of mechanization of appropriate manual tasks on a broad front.

Signal surveillance conducted by the armed forces, recognized as an essential and integral part of intelligent tactical and strategic planning, is one such area where automation is receiving increased attention and support. In particular, the human operator has long been relied upon to provide the necessary manual transcription of manual morse circuits under surveillance. Because of the reduced manpower levels, this surveillance and transcription must be tranferred to mechanized equipment if this source of intelligence is to remain timely and effective.

This thesis investigates the problem of automatic transcription of the morse signal, and describes and documents

several approaches to filtering, processing, and decoding it for transcription. The baseband morse signal process is first modeled as a modified random telegraph wave. A discrete Kalman filter and a linear smoother are then used to process the demodulated signal in order to gain a measure of the effectiveness and applicability of this model. It is shown experimentally that this model and processing yield a significant reduction in the transcription error rate. Next, a Viterbi decoder algorithm based on a simple Markov model of the code is programmed and tested. Finally, a more general model of the signal process, incorporating the baseband model, is used to design and implement a pre-detection Kalman filter.

## II. PROBLEM DESCRIPTION

### A. THE MANUAL MORSE SIGNAL PROCESS

It is assumed that the reader is familiar with the manual morse signal, its pecularities, vagaries, and uncertainties, and with current methods of transcription. To formalize the discussion, however, certain definitions of the terms used and a brief description of the signal are in order.

As used throughout this report, the term morse signal refers to International Morse Code, sent manually by key, manual "bug", or electronic keyer. The problem of transcribing keyboard morse, that which is sent automatically with standard parameters, will not be considered, although certain results are applicable. The baseband morse signal is the output of the keyer and is represented by the logic levels "0" and "1", corresponding to the states "key up" and "key down." The five characters of the international morse code are identified as: dot, dash, element-space, letter-space, and word-space. The term element[1] refers to the standard time unit of the code; its actual duration in seconds will of course vary with sending speed. Standard morse code consists of the character lengths shown in Table I.

---

[1]Sometimes the terms baud and bit are used.

TABLE I

STANDARD MORSE CHARACTERS

| Character | Symbol | Duration (in elements) |
|---|---|---|
| Dot | . | 1 |
| Dash | _ | 3 |
| Element-space | ^ | 1 |
| Letter-space | ~ | 3 |
| Word-space | w | 7 |

The standard word (including word-space) in morse communication is 50 elements in length. Thus the element duration in seconds for a given sending speed may be calculated as 6/5 times the reciprocal of the speed in words per minute. The author is unaware of any generally accepted standard for the bandwidth of the baseband morse signal; it was found to be convenient to express the upper limit of the bandwidth as three times the reciprocal of the element duration. Thus a code speed of 36 wpm has a bandwidth of 90 Hz.

An actual (as opposed to standard) morse signal, as those familiar with the problem are aware, may exhibit quite a wide variation from standard code in character duration, speed variability, and consistency of element duration. Since these variations are often unique to the particular sending operator, and in many cases may depend

on the type of traffic being sent as well, it is difficult, if not impossible, to describe a "typical" morse signal. This variability is illustrated in Figures 1 and 2, which are histograms of the character duration of two different amateur ratio operators recorded on the air. As can be seen, the distributions are different, with the least variability in both cases appearing in the dot and element-space durations.

## B. SYSTEM CONSTRAINTS

The signal processor will obviously play an important role in the automated manual morse intercept and transcription system. In order to fully appreciate the system constraints under which the processor was developed, and the context in which it is expected to operate, an outline of the integrated system is presented. Referring to the system block diagram, Figure 3, its three basic components may be briefly described as follows.

### 1. Modulation Sorting Subsystem

This subsystem scans the band and/or frequencies of interest and detects the presence of morse signals in a (typically) 2 kHz band. Upon detection of a morse signal, a separate digitally-tuned receiver is automatically tuned to the signal frequency for reception.

### 2. Morse Processor

This signal processor is currently the only part of the system which is not in existing hardware or software.

Its basic function is to minimize error probability in the face of noise, interference, and uncertain signal parameters.

3. Automatic Transcriber

This component translates the code into letters of the alphabet; there are currently several transcribers available which have proven effective at adequate signal-to-noise ratios with modest signal parameter variation [1],[2].

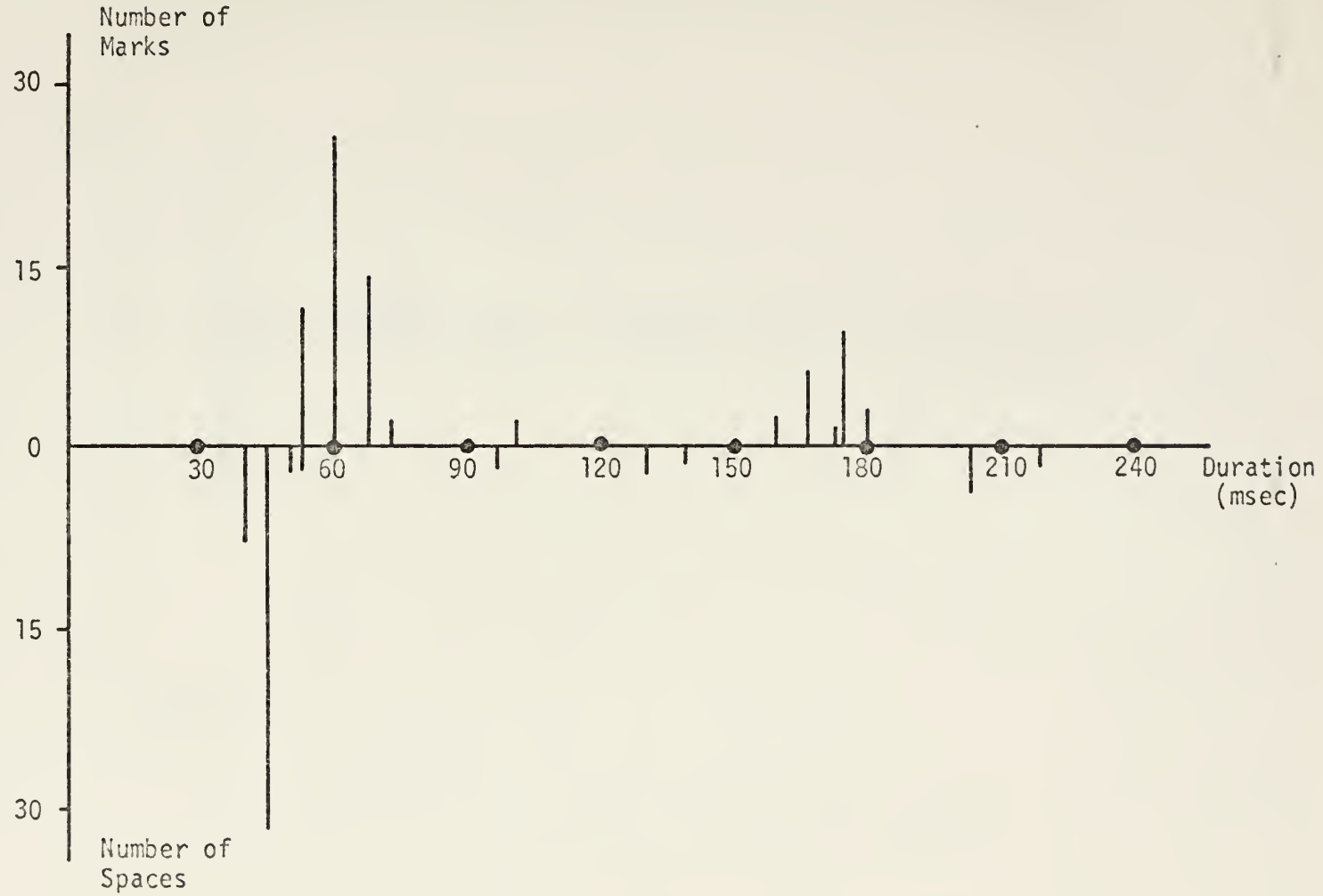



Figure 1. Character Duration Histogram 1

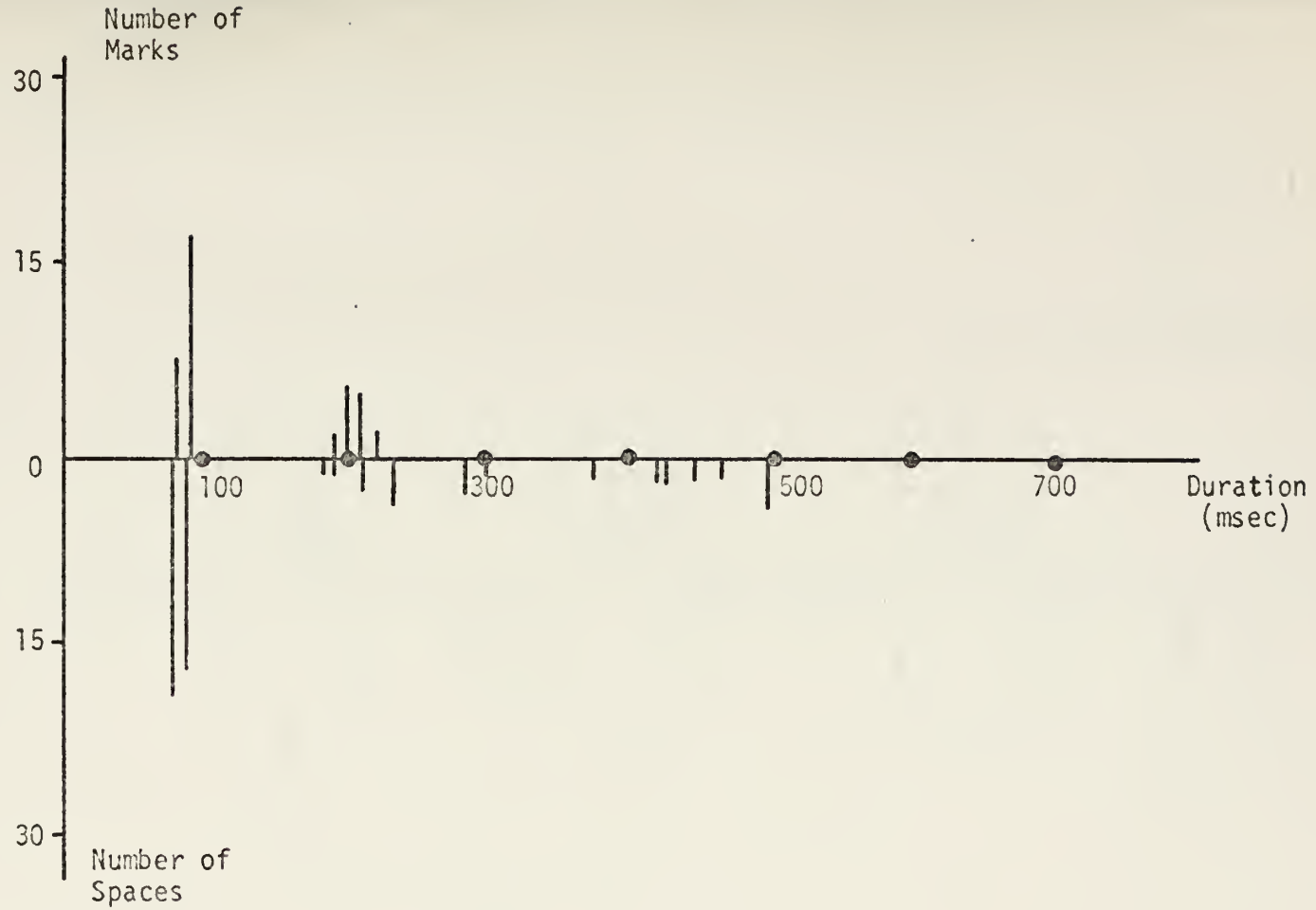



Figure 2. Character Duration Histogram 2

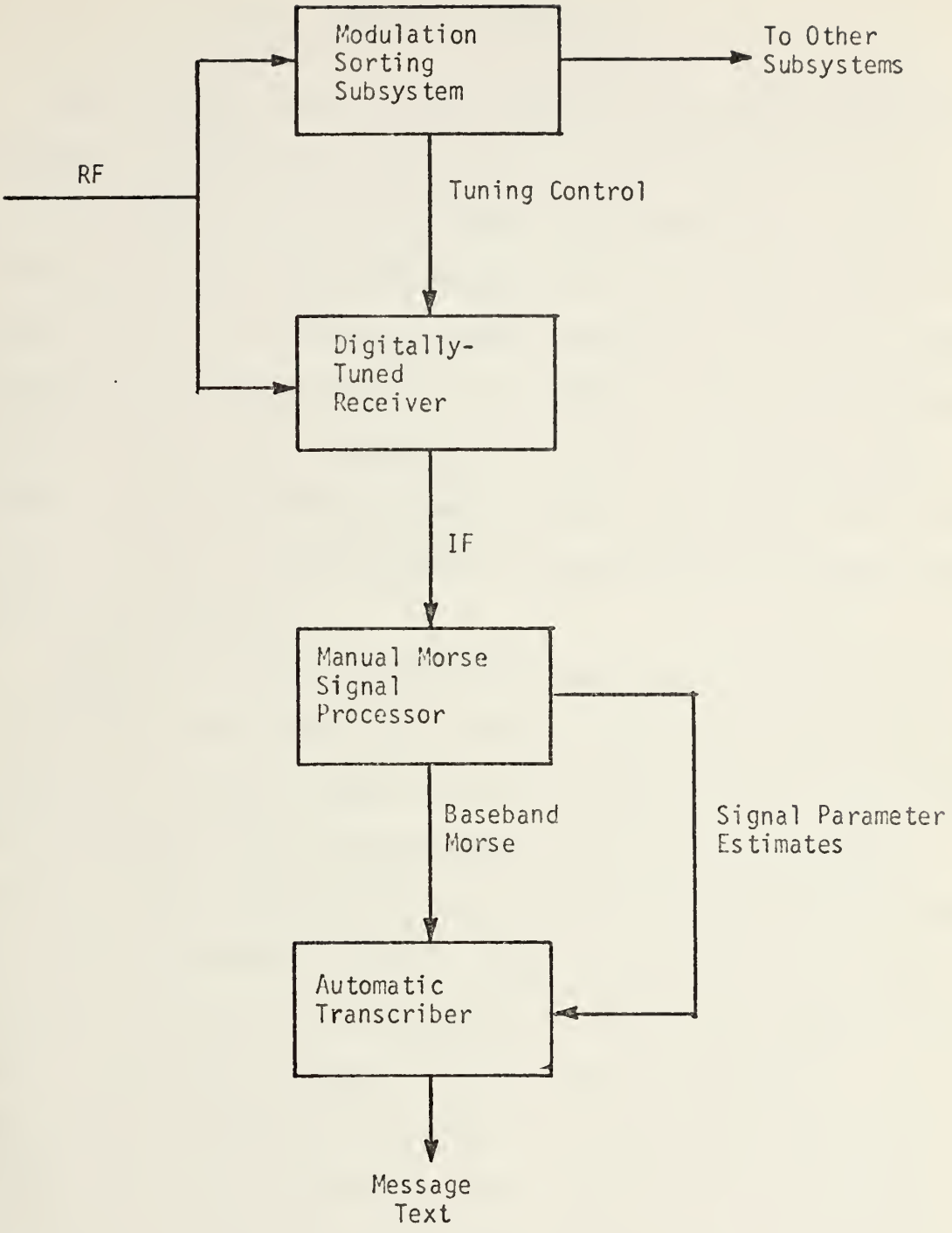



Figure 3. System Block Diagram

## III. DESIGN OBJECTIVES

The ultimate goal of the intercept and transcription system is to provide output copy with an error rate no greater than that which a "good" human operator can provide. Such an operator manually transcribing the morse signal can adapt rapidly to changing signal parameters and has little trouble distinguishing dots from dashes even if the sender's keying is far from perfect. Additionally, he can adapt readily to the noise and interference environment and reliably copy a signal in the presence of numerous other morse and non-morse signals.

Specific operator performance data were not available to the author, thus only broad design objectives were formulated based on a limited number of both subjective and experimental data obtained using amateur radio operators as subjects. Random letter sequences were sent using the Pickering model KB-1 morse keyboard to key a signal generator at an audio frequency selected by the subject. Noise was added to the audio signal and the SNR in each bandwidth used was recorded. The results, summarized in Table II, tabulate error rate versus SNR in the bandwidth used. Also shown is the SNR in the signal bandwidth as previously defined in Section II.A.

The conclusions drawn from these data is that a good operator can copy reasonably well down to -13 dB SNR in a 2 kHz bandwidth. Although the insertion of a 100 Hz bandpass filter raises the SNR to 0 dB, the relative invariance of

TABLE II

OPERATOR PERFORMANCE DATA

(a) Speed: 35 wpm

| BW | SNR (dB) (in given BW) | SNR (dB) (in signal BW) | ERROR RATE (%) OP1 | OP2 | COMMENTS |
|---|---|---|---|---|---|
| 2 kHz | -13 | 0 | 12 | 15 | difficult and |
| 200 Hz | -3 | 0 | 11 | 13 | fatiguing |
| 100 Hz | 0 | 0 | 12 | 14 | BW too narrow |
| 2 kHz | -10 | 3 | 6 | 10 | fatiguing |
| 200 Hz | 0 | 3 | 11 | 9 | BW too narrow |
| 100 Hz | 3 | 3 | 11 | 10 | |
| 2 kHz | -7 | 6 | 1 | 2 | relatively easy |
| 200 Hz | 3 | 6 | 0 | 1 | prefer wider BW |
| 100 Hz | 6 | 6 | 2 | 2 | |

(b) Speed: 25 wpm

| BW | SNR (dB) (in given BW) | SNR (dB) (in signal BW) | ERROR RATE (%) OP1 | OP2 | COMMENTS |
|---|---|---|---|---|---|
| 2 kHz | -13 | 2 | 6 | 5 | difficult and |
| 200 Hz | -3 | 2 | 7 | 6 | fatiguing |
| 100 Hz | 0 | 2 | 8 | 5 | |
| 2 kHz | -10 | 5 | 2 | 1 | relatively easy |
| 200 Hz | 0 | 5 | 1 | 1 | but still mildly |
| 100 Hz | 3 | 5 | 2 | 2 | fatiguing |
| 2 kHz | -7 | 8 | 1 | 0 | easy enough |
| 200 Hz | 3 | 8 | 0 | 1 | |
| 100 Hz | 6 | 8 | 0 | 2 | |

error rates with changes in filter bandwidth indicates that the ear performs the necessary filtering. Strict concentration is required, however, at this low SNR, and the test operators stated they would not attempt to copy such a signal unless strongly motivated.

Using the previously defined signal bandwidth for each speed, the results may be summarized as follows: An operator can provide copy with a 10-15% error rate with an SNR of approximately 0 dB in the signal bandwidth; at 3 dB, the error rate is 5-10%; and at 6 dB, copy is practically perfect.

Based on these results, it seems reasonable to assert that a typical field operator, faced with searching for and copying morse traffic eight hours a day, would not be inclined to copy signals much below 6 dB SNR in the signal bandwidth unless absolutely required. Thus the system designer must select from two costly alternatives: 1) If he designs the system to perform as well as the <u>good</u> operator <u>is able</u> to perform, the automated system will reliably receive a large percentage of signals encountered on the air, but it is likely to be complex and expensive. 2) On the other hand, if he designs the system to perform as well as a <u>typical</u> operator probably performs, the automated system will be cheaper, but the remaining operators who must copy the low SNR signals which are not machine transcribable may become too fatigued to be efficient, leaving the overall man/machine surveillance system less effective than the existing manual system.

Such design considerations are beyond the scope of this thesis; using the "good" operator as a criterion, the

ultimate system design objectives may be broadly stated as follows:

1) With an error rate of 10% or less, recover and decode morse signals whose SNR in a 2 kHz bandwidth is on the order of -10 dB, using standard code with additive white gaussian noise and no interference.
2) Track the time-varying statistics of character lengths in order to enable the transcriber to translate the code properly.

## IV. PROCESSOR DESIGN PHILOSOPHY

The first step in the processor design was to model the morse code as a random telegraph wave with non-stationary transition probabilities. Using this model, several increasingly complex processing methods were implemented, and the processing gain of each stage was determined. First a Kalman filter was designed to filter the demodulated output of a square-law detector with and without narrowband analog IF filtering. Next a smoothing algorithm was added to determine the effectiveness over Kalman filtering alone. Finally a maximum a posteriori (MAP) estimator of the code characters, using the smoothed output for likelihood calculation, and using the Viterbi algorithm for processing, was programmed and tested.

After determination of the error-reduction effectiveness of each of these processing stages, a more general model of the signal process was used to design a Kalman filter for pre-detection filtering. The objective was to determine whether or not such filtering yielded any advantage over the simpler demodulation/post-detection filter approach. A block diagram of the various stages is shown in Figure 4. Sections V through VII present a theoretical basis for each of the processing stages, followed by a presentation and discussion of experimental results.

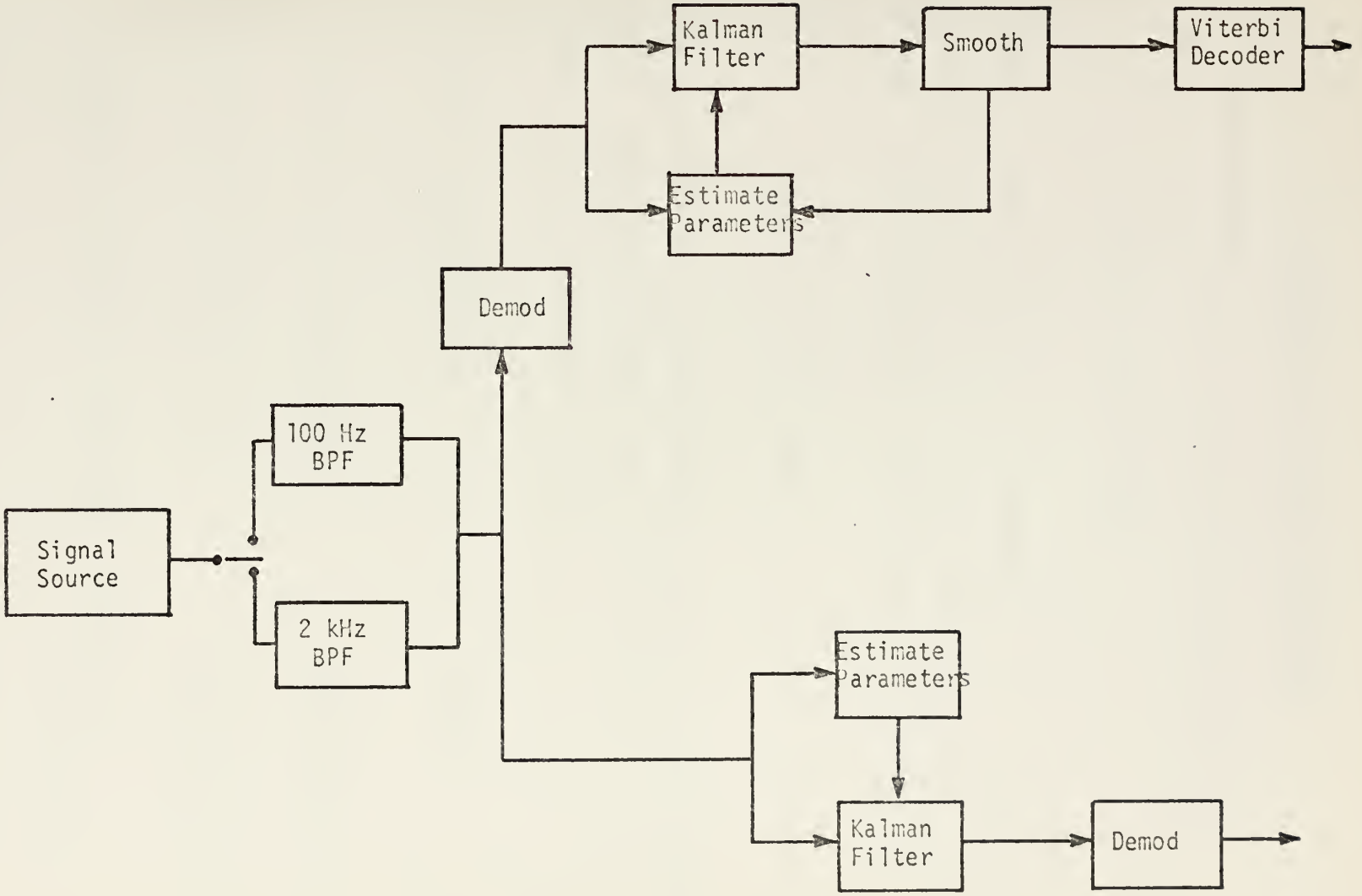



Figure 4. Processor Design Stages

## V. BASEBAND MODEL AND PROCESSOR

The baseband morse signal may be modeled as

$$x(k+1) = x(k) + w(k) \qquad \text{(V-1)}$$

where x = key state (0 or 1), and w(k) is a random forcing function descriptive of the morse keying process. At the output of the demodulator, the signal is observed as

$$z(k) = x(k) + v(k)$$

where z(k) is the observed value and v(k) is the additive noise. This model gives rise to the following scalar Kalman filter algorithm [3],[4]:

$$G(k) = \frac{V(k|k-1)}{V(k|k-1) + R} \qquad \text{(gain)}$$

$$V(k|k) = [1 - G(k)]V(k|k-1) \qquad \text{(estimation variance)}$$

$$V(k+1|k) = V(k|k) + Q(k) \qquad \text{(prediction variance)}$$

$$\hat{x}(k|k) = \hat{x}(k|k-1) + G(k)[z(k) - \hat{x}(k|k-1)] \qquad \text{(estimation)}$$

$$\hat{x}(k+1|k) = \hat{x}(k|k) \qquad \text{(prediction)}$$

where:

$\hat{x}(k)$ = estimate of x at time k

R = variance of the observation noise, v(k)

Q(k) = variance of the random forcing function, w(k).

Since this algorithm implies knowledge of the variances R and Q, procedures for their estimation are required. In order to keep the filter algorithm itself as simple as possible, estimates of Q and R were made independently, and used by the filter algorithm as if these were the true values.

## A. ALGORITHM FOR ESTIMATING THE RANDOM FORCING FUNCTION VARIANCE

The random forcing function w(k) is descriptive of the on-off keying process, i.e., it may be thought of as the process which causes the transitions in the x state from 0 to 1 and from 1 to 0. Referring to the signal model equation (V-1), the keying process has the following interpretation: If x(k) = 0 and w(k) = 0, then x(k+1) = 0 and x remains in the space condition. If, on the other hand, w(k) = 1, x shifts (at time k+1) from the space condition to the mark condition. This process is illustrated in Figure 5. Since the probability of occurrence of a transition is dependent on the time duration since the last transition, this probability is non-stationary. A proper description of the transition probabilities at each time k, then, must necessarily be time dependent and conditioned on the element

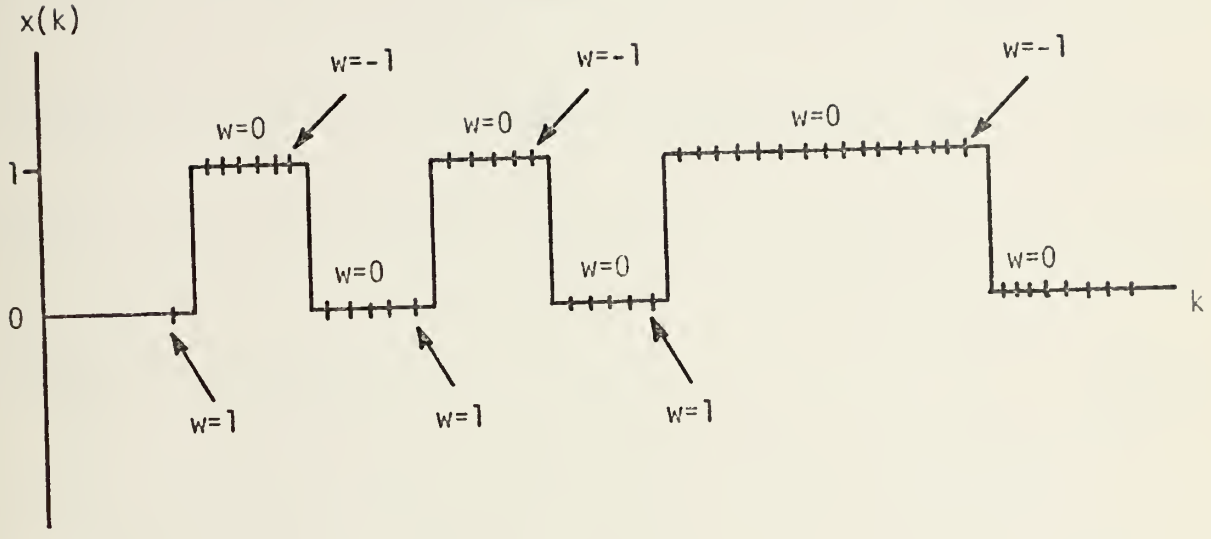



Figure 5. Illustration of Signal Model Process

duration and the present state. The following probabilities are therefore required:

$$P(w(k)=0|k,x(k)=1) = \Pr[\text{remain in mark condition}] \quad \text{(V-2)}$$

$$P(w(k)=1|k,x(k)=1) \equiv 0 \quad \text{(V-3)}$$

$$P(w(k)=-1|k,x(k)=1) = \Pr[\text{transition from mark to space}] \quad \text{(V-4)}$$

$$P(w(k)=0|k,x(k)=0) = \Pr[\text{remain in space condition}] \quad \text{(V-5)}$$

$$P(w(k)=1|k,x(k)=0) = \Pr[\text{transition from space to mark}] \quad \text{(V-6)}$$

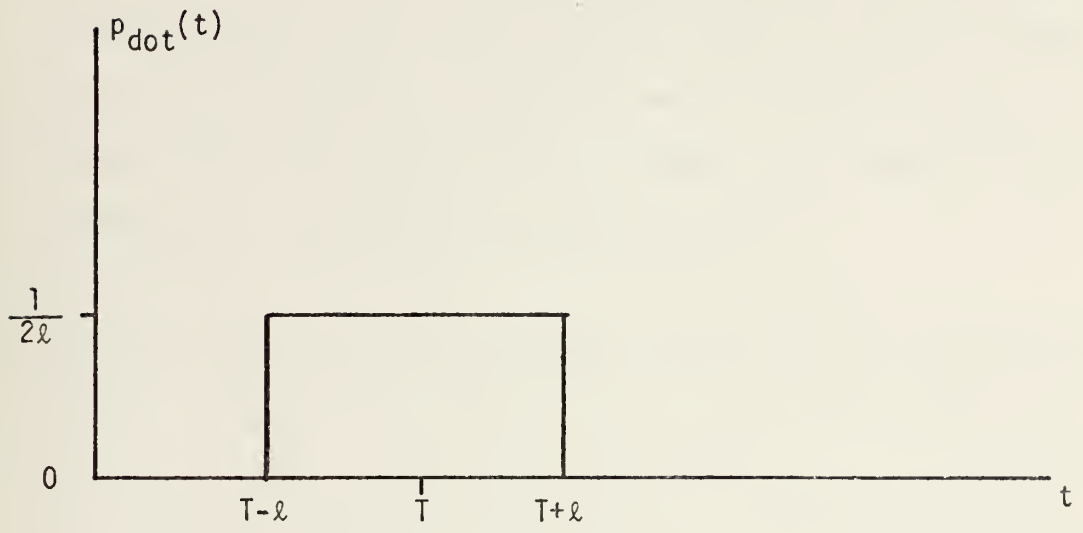
$$P(w(k)=-1|k,x(k)=0) \equiv 0 \quad \text{(V-7)}$$

Probabilities (V-3) and (V-7) are identically zero since state values less than 0 and greater than 1 are not allowable. The remaining probabilities are dependent on the distributions of dot, dash, element-space, letter-space, and word-space durations of the particular morse signal being received, and are dependent in a Markov sense on the previous character. The Markov nature of the code character transitions was not taken advantage of in the filtering process. Additionally it was assumed that a particular operator's character

durations are all distributed with uniform density, that the dot and element space have the same density, and that the dash and letter space have the same density. The assumed densities, as shown in Figure 6, presume for the present that the mean values T, $T_d$, and the parameters $\ell$, d are either known or have been determined in some manner.

Although the assumption of uniform densities for the character durations of any particular sending operator is probably not strictly justifiable, neither is the assumption of any other well-known density, such as a gaussian or exponential density. The most likely candidate for properly modeling these duration distributions may be a gaussian-like density with the tails truncated at suitable values. The complexity of estimating the parameters of such a density for a particular received signal, together with the computational burden of evaluating the error function, erf(t), for the probability calculations (V-2) through (V-7) at each sample point, motivated the selection of the uniform density. The resulting probability computations are relatively simple and straightforward.

a) Dot/Element-space Duration Density




b) Dash/Letter-space Duration Density

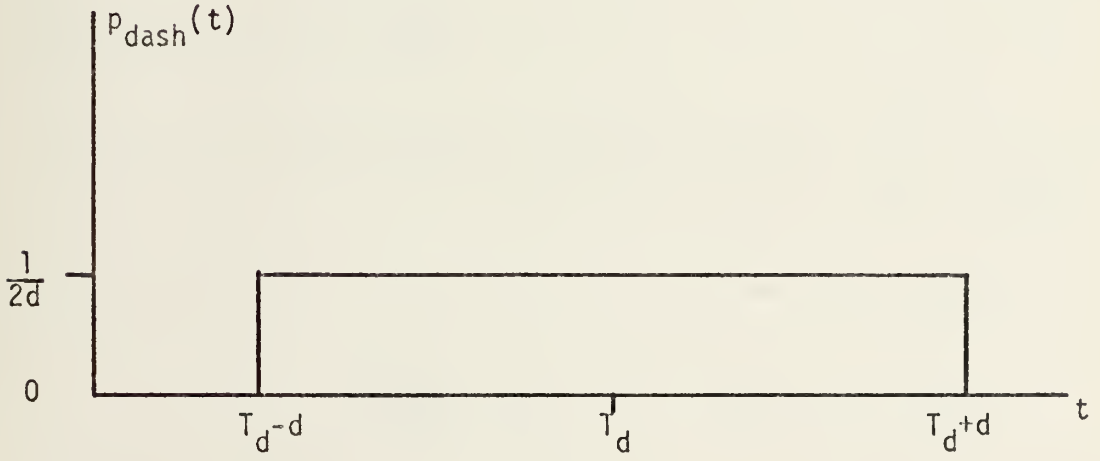



Figure 6. Character Duration Densities

1. Mark Transition Probabilities

A description of probabilities (V-2) and (V-4) may be obtained by first conditioning on the dot and dash probabilities, and noting that if x(k) is still in the mark condition after T+ℓ, then the mark is known to be a dash. Given then that a mark, m, is a dot, and given T and ℓ, probability (V-2) may be modeled as

$$P_{01}(\text{dot}) \triangleq P(w(k)=0 \,|\, k,T,x(k)=1,m=\text{dot}) = \int_k^{T+\ell} p_{\text{dot}}(t)\, dt$$

$$= \begin{cases} 1 \; ; & 0 \leq k \leq T-\ell \\ \frac{T-k}{2\ell} + \frac{1}{2} \; ; & T-\ell \leq k \leq T+\ell \\ 0 \; ; & T+\ell \leq k \end{cases}$$

Similarly, for a dash:

$$P_{01}(\text{dash}) \triangleq P(w(k)=0 \,|\, k,T_d,x(k)=1,m=\text{dash}) = \int_k^{T_d+d} p_{\text{dash}}(t)\, dt$$

$$= \begin{cases} 1 \; ; & 0 < k \leq T_d-d \\ \frac{T_d-k}{2d} + \frac{1}{2} \; ; & T_d-d \leq k \leq T_d+d \\ 0 \; ; & T_d+d \leq k \end{cases}$$

Here k is the time index which counts the number of units the signal has been in the mark condition on the present mark. The conditioning on m must be removed for $0 \leq k \leq T+\ell$ since m is not known during this interval. Thus for $k \leq T+\ell$:

$$P(w(k)=0 \mid k,T,x(k)=1) = P_{01}(\text{dot})\Pr(\text{dot}) + P_{01}(\text{dash})\Pr(\text{dash}) \qquad \text{(V-8)}$$

and for $k > T+\ell$

$$P(w(k)=0 \mid k,T_d,x(k)=1) = P_{01}(\text{dash}) \; . \qquad \text{(V-9)}$$

But the dot and dash probabilities are dependent on the type of traffic in the message, i.e. on what language the message is in, whether it is plain text or code groups, letters only or both letters and numbers, etc. Since the traffic type may not be known a priori, equiprobable dots and dashes were assumed at this point. Using Pr(dot) = Pr(dash) = 1/2, then, equations (V-8) and (V-9) reduce to

$$P(w(k)=0 \mid k,T,T_d,x(k)=1) = \begin{cases} 1 & 0 \leq k \leq T-\ell \\ \frac{T-k}{4\ell} + \frac{3}{4} & T-\ell \leq k \leq T+\ell \\ 1 & T+\ell < k \leq T_d-d \\ \frac{T_d-k}{2d} + \frac{1}{2} & T_d-d \leq k \leq T_d+d \\ 0 & T_d+d \leq k \end{cases}$$

Probability (V-6) follows immediately as

$$P(w(k)=-1|k,T,T_d,x(k)=1) = 1 - P(w(k)=0|k,T,T_d,x(k)=1) \ .$$

The expression for probability (V-2) as a function of time, k, conditioned on the present state and dot and dash distributions is sketched in Figure 7.

2. Space Transition Probabilities

An appropriate description for the space duration probabilities (V-4) and (V-7) is derived similarly, first by conditioning on a particular space and then removing the appropriate conditioning by using the relative frequencies of each space. Given that a particular space, s, is an element-space, then,

$$P(w(j)=0|x(j)=0,s=elem,T) = \begin{cases} 1 & 0 < j \leq T-\ell \\ \frac{T-j}{2\ell} + \frac{1}{2} & T-\ell \leq j \leq T+\ell \\ 0 & T+\ell \leq j \end{cases} \qquad \text{(V-10)}$$

where j is the time index which counts the number of units the signal has been in the space condition on the present space; and the element length, given T, is assumed to have the same uniform density as the dot length.

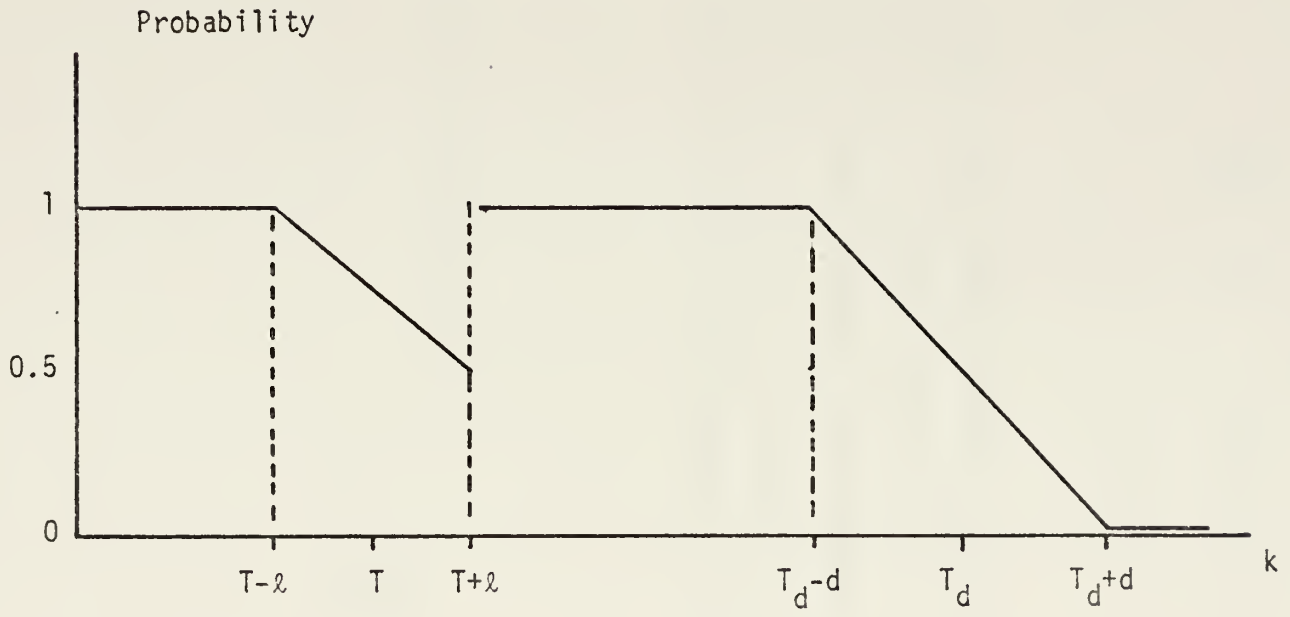



Figure 7. Mark Probability (V-2) as Function of Time Index

Similarly for a letter-space:

$$P(w(j)=0 \mid x(k)=0, s=\text{ltr-sp}, T_d) =$$

$$\begin{cases} 1 & 0 < j \leq T_d - d \\ \dfrac{T_d - j}{2d} + \dfrac{1}{2} & T_d - d \leq j \leq T_d + d \\ 0 & T_d + d \leq j \end{cases} \qquad \text{(V-11)}$$

where j is the same index as above and the letter-space length, given $T_d$, has the same density as the dash-length.

For the word-space it was decided to use an exponential model, since after about 5T units, the word-space is (in actual practice) about equally likely to end at any time:

$$P(w(j)=0 \mid x(j)=0, s=\text{word-sp}, T) = \begin{cases} 1 & 0 < j \leq 5T \\ e^{-\left(\frac{j-5T}{2T}\right)} & 5T \leq j < \infty \end{cases} \qquad \text{(V-12)}$$

The conditioning on s may be removed by observing that if $j > T_d+d$, the space must be a word-space; if $j > T+\ell$, then the space is a word-space or letter-space. In equiprobable letter traffic, the needed probabilities are:

$$\Pr(s=\text{elem-space}) = 12/17$$

$$\Pr(s=\text{letter-space}) = 4/17$$

$$\Pr(s=\text{word-space}) = 1/17$$

Applying these probabilities then to (V-10), (V-11), and (V-12) yields the desired expression:

$$P(w(j)=0 \mid j,T,T_d,x(j)=0) =$$

$$\begin{cases} 1 & 0 < j \leq T-\ell \\ [\frac{T-j}{2\ell} + \frac{1}{2}] \cdot \frac{12}{17} + \frac{5}{17} \quad ; & T-\ell \leq j \leq T+\ell \\ 1 & T+\ell < j \leq T_d-d \\ [\frac{T_d-j}{2d} + \frac{1}{2}] \cdot \frac{4}{5} + \frac{1}{5} \quad ; & T_d-d < j \leq T_d+d \\ 1 & T_d+d < j \leq 5T \\ e^{-(\frac{j-5T}{2T})} & 5T \leq j \end{cases}$$

3. Calculation of Variance

The variance of w(k) is now easily calculated at each time, k, as follows:

Let

$$P_k(0,1) \triangleq \Pr(w(k)=0 \mid x(k)=1)$$

$$P_k(-1,1) \triangleq \Pr(w(k)=-1 \mid x(k)=1)$$

$$P_j(0,0) \triangleq \Pr(w(j)=0 \mid x(j)=0)$$

$$P_j(1,0) \triangleq \Pr(w(j)=1 \mid x(j)=0) \quad .$$

Then, if x(k) = 1,

$$Q(k) = (0)^2 \cdot P_k(0,1) + (-1)^2 \cdot P_k(-1,1) - [E(w(k))]^2$$

$$= P_k(-1,1) - [-P_k(-1,1)]^2$$

$$= P_k(-1,1) \cdot P_k(0,1) \quad .$$

Similarly, if x(j) = 0,

$$Q(j) = P_j(0,0) \cdot P_j(1,0) .$$

Since, at the receiver, the true state x(k) is not known, correct estimates of Q are dependent on correct estimates of the state, $\hat{x}(k)$. Thus, it is expected that at some SNR, incorrect estimates of x(k) will cause the

estimates of Q to be erroneous enough to force incorrect estimates of x(k+1), causing a runaway condition to develop and yield the receiver worthless. This SNR at which runaway develops was determined experimentally, and found not to be a serious problem.

B. CHARACTER DISTRIBUTION ESTIMATION

It was assumed that the character distributions are appropriately described by uniform densities with known parameters (expect for mean values). More sophisticated methods of distribution and/or parameter estimation were discarded in favor of simplicity. Experience suggests that even a sloppy sender will usually not exceed an $\ell/T$ or $d/T_d$ ratio of about 1/3. Thus, once estimates of T and $T_d$ are obtained, $\ell$ and d can be determined from this assumption unless these ratios are known in advance. The mean values T and $T_d$ of dot or element-space and dash or letter-space, respectively, were estimated by measuring the character durations and sequentially averaging the appropriate duration. Thus T is the mean value of the dot and element-space durations, and $T_d$ is the mean value of the dash and letter-space durations. The threshold for deciding which set of measurements a particular length belongs to was set at the halfway-point between dot and dash lengths. The algorithm is as follows:

Initially specify:

$t_1$ = shortest dot-duration expected,

$t_2$ = longest dot-duration expected,

with $t_2 \leq 3t_1$.

1. (a) Initialize estimate of T as $(t_1+t_2)/2$.

   (b) Initialize estimate of $T_d$ as $(3t_1+3t_2)/2$.

2. Measure the duration of each mark and space.

3. If the measured value is less than $(3t_1+t_2)/2$, then identify it as a dot/element duration, $T_1$.

4. If the measured value is larger than $(3t_1+t_2)/2$, then identify it as a dash/letter-space duration, $T_2$.

5. Estimate the means recursively by

   (a) $\hat{T}(k) = \hat{T}(k-1) + \frac{1}{k} [T_1 - \hat{T}(k-1)]$ .

   (b) $\hat{T}_d(k) = \hat{T}_d(k-1) + \frac{1}{k} [T_2 - \hat{T}_d(k-1)]$ .

C. OBSERVATION NOISE VARIANCE

Although the noise power used in each of the tests was known and could have been specified initially, it was decided to estimate this parameter in order to better simulate an operational environment where the noise power is not known a priori. Although the method is purely intuitive and without a valid theoretical basis, reasonably good results were obtained.

At the output of the square-law demodulator, the noise is no longer gaussian and is correlated with the signal. Proceeding, however, as if the signal and noise were not correlated, the noise variance R can be obtained by subtracting the morse signal power from the total received (demodulated) signal power:

$$\hat{R} = \hat{V}_z - \frac{\hat{a}^2}{4}$$

where $\hat{V}_z$ is obtained recursively by the following:

$$\hat{V}_z(k) = \hat{V}_z(k-1) + \frac{1}{k}\left[(z-\hat{\mu}_z)^2 - \hat{V}_z(k-1)\right]$$

and

$\hat{\mu}_z$ = mean value of the received (demodulated) signal

$\hat{a}$ = estimate of demodulated signal amplitude.

The parameters $\hat{\mu}_z$ and $\hat{a}$ were also estimated on line, although they too are known a priori for test purposes. The estimator algorithm for $\hat{a}$ is

1) $\hat{a}_1(k) = \hat{a}_1(k-1) + \frac{1}{k}\left[z(k) - \hat{a}_1(k-1)\right]$ if $z(k) \geq \hat{\mu}_z$

(2) $\hat{a}_2(k) = \hat{a}_2(k-1) + \frac{1}{k}\left[z(k) - \hat{a}_2(k-1)\right]$ if $z(k) < \hat{\mu}_z$

(3) $\hat{a}(k) = \hat{a}_1(k) - \hat{a}_2(k)$ .

The mean value, $\hat{\mu}_z$, is simply

$$\hat{\mu}_z(k) = \hat{\mu}_z(k-1) + \frac{1}{k}\left[z(k) - \hat{\mu}_z(k-1)\right] .$$

D. SMOOTHING ALGORITHM

The Kalman-filtered output represents the optimum linearly-filtered estimate of the signal amplitude in the minimum mean-square-error sense based on the assumed model and the efficiency and consistency of the estimated statistics. This estimate may be improved, however, by smoothing the data, which implies consideration of future inputs as well as past inputs [4].

The equations for the discrete optimal linear fixed-interval smoother, for the scalar case, expressed as a combination of a forward-running filter and backward-running filter are given by [5],[6] as:

Smoothed Estimate:

$$\hat{x}(k|N) = P(k|N)\cdot[\hat{x}(k|k)/P_f(k|k) + \hat{x}_b(k|k+1)/P_b(k|k+1)]$$

Estimation Variance:

$$P(k|N) = \frac{1}{1/P_f(k|k) + 1/P_b(k|k+1)} \quad ,$$

where

$\hat{x}(k|k)$ = filtered state estimate

$P_f(k|k)$ = error variance for the forward filter

$\hat{x}_b(k|k+1)$ = (predicted) estimate for the backward filter

$P_b(k|k+1)$ = (predicted) error variance, backward filter.

These expressions are not computationally suitable, however, since $P_b(k|k+1)$ is not finite when $k = N$. Alternate expressions [6] which are more amenable to computation are given as follows:

Let $$W(k) = \frac{P_f(k|k)}{1 + P_f(k|k)/P_b(k|k+1)} .$$

Then

$$P(k|N) = [1 - W(k)/P_b(k|k+1)]^2 \cdot P_f(k|k) \cdot [W^2(k)/P_b(k|k+1)]$$

and

$$1/P_b(k|k+1) = 0 \quad \text{when} \quad k = N .$$

Let $$W_b(k|k+1) = x_b(k|k+1)/P_b(k|k+1) .$$

Then

$$\hat{x}(k|N) = \frac{\hat{x}(k|k)}{[1 + P_f(k|k)/P_b(k|k+1)] + P(k|N)W_b(k|k+1)}$$

and

$$P(k|N)W_b(k|k+1) = 0 \quad \text{when} \quad k = N .$$

These equations then represent the smoothing algorithm. As noted above, forward-filtered estimates must be stored until the same data can be backward-filtered and combined to produce the smoothed state estimate.

The smoothed state estimate then is the best (linear) estimate possible for the given model and parameter estimators, and any improvement in error rate must be derived from giving more probabilistic structure to the signal model.

## E. IMPLEMENTATION OF FILTER AND SMOOTHER

A diagram of the test signal generation method is shown in Figure 8. The square-law demodulator was selected simply for ease of implementation since an analog squarer and appropriate filters were readily available and easily interfaced with the analog-to-digital converter. The 100 Hz low-pass filter permits recovery of morse signals of approximately 35 wpm or less. The signal-to-noise ratios used throughout this report are <u>average</u> (pre-detection) signal-to-noise power ratios and not <u>pulse</u> signal power to noise power ratios. More specifically, the average signal power in the morse signal is defined as

$$P_s = \frac{1}{2T} \int_0^{2T} [s(t)]^2 \, dt$$

where the interval [0,2T] is a dot and element-space. This expression reduces to

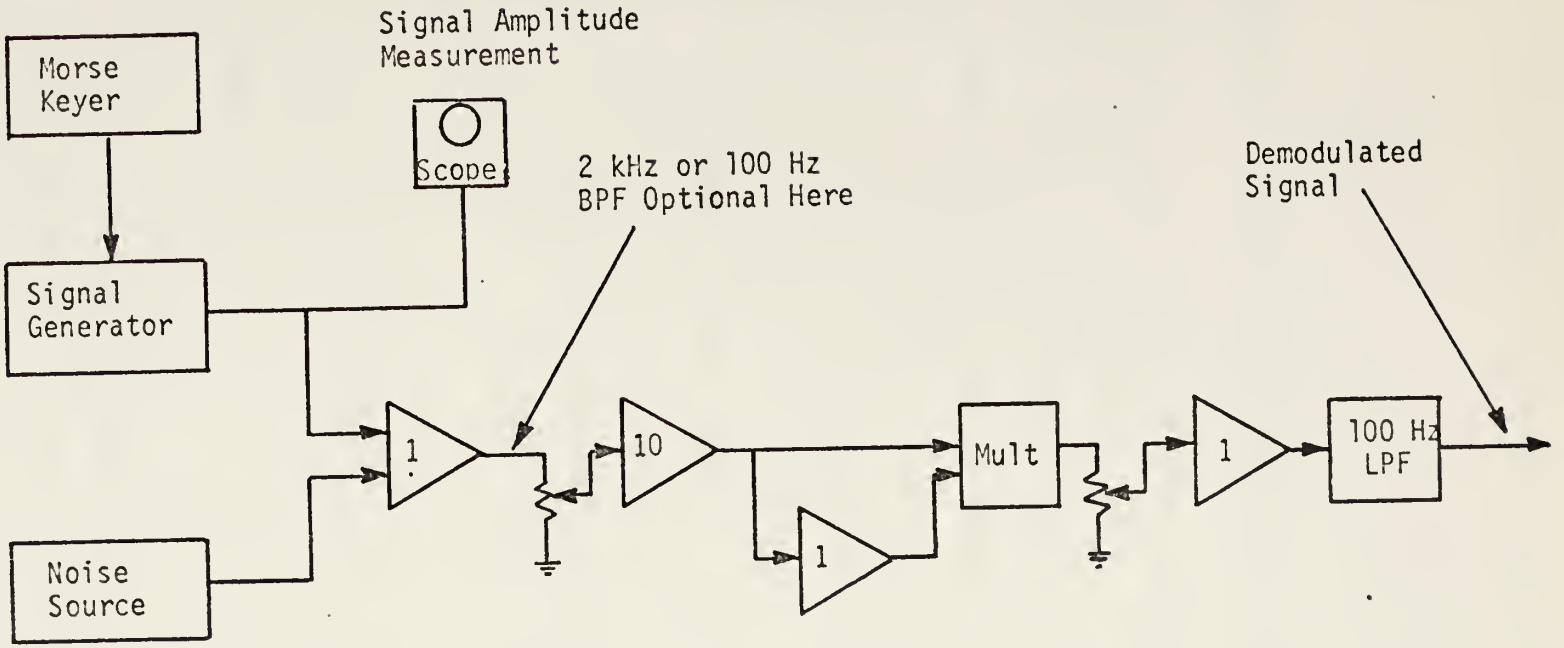



Figure 8. Signal Generation and Demodulation Block Diagram

$$P_s = \frac{1}{2T} \int_0^T (a \cos \omega t)^2 \, dt + \frac{1}{2T} \int_T^{2T} 0 \, dt$$

$$P_s = a^2/4 \quad ,$$

which is 3 dB less than the pulse signal power.

The noise signal is taken from a calibrated white gaussian noise source of 1 volt rms.

The filter and smoother, along with the auxiliary estimation algorithms, were coded in Fortran and implemented on the XDS-9300 computer interfaced with the CI-5000 analog computer for analog filtering and for D/A and A/D conversion. The sampling rate was 500 samples per second, and the value of N for smoothing was chosen to be 250 samples. The sampled test signal was recorded on tape for subsequent processing, since the processing required approximately 4 seconds for 1 second of data. The test signal runs consisted of the following:

1. Perfect code $\overline{AR}$ sequence and random letter sequence at speeds of 35, 30, and 25 wpm each with a signal-to-noise ratio of 6, 5, 4, 3, 2, and 1 dB in a 2 kHz BW with no pre-detection analog filtering.
2. Perfect code $\overline{AR}$ sequence and random letter sequence at speeds of 35, 30, and 25 wpm each with a signal-to-noise ratio of -7, -8, -9, -10, -11, -12 dB in a 2 kHz BW with a 100 Hz pre-detection analog bandpass filter.

3. Sloppy code $\overline{AR}$ sequence at nominal speeds of 25 and 30 wpm each with a signal-to-noise ratio of 6, 5, 4, 3, 2, 1 dB in a 2 kHz BW with no pre-detection filtering.
4. Sloppy code $\overline{AR}$ sequence at nominal speeds of 25 and 30 wpm each with a signal-to-noise ratio of -7, -8, -9, -10, -11, -12 dB in a 2 kHz BW with a 100 Hz pre-detection analog bandpass filter.

The output of the processor was recorded on an 8-channel strip-chart, using a utility D/A conversion routine. The outputs were as follows:

Channel 1: Recorded input signal

Channel 2: Unprocessed output (input signal thresholded at its mean value)

Channel 3: Kalman filtered output

Channel 4: Filtered output thresholded at 0 V.

Channel 5: Smoothed output

Channel 6: Smoothed output thresholded at 0 V.

Channel 7: Option of filter gain or noise variance estimate

Channel 8: Dot/element-space duration estimate.

The following figures (9-19) are typical output records, showing examples of the test runs for each signal and type of sequence. Channels 1-4 are shown in Figure (a) in each case with the corresponding channels 5 and 6 shown in Figure (b). An example of the output of channel 7 for the gain option is shown in Figure 20, along with the corresponding outputs from channels 1, 3, and 4. Figure 21 shows an

example of the outputs for the noise variance estimate and the dot/element-space duration estimate at a point where the input signal changes from 25 to 30 wpm and the signal-to-noise ratio changes from 1 dB to 6 dB. In all cases the chart speed was 10 mm/sec except in Figure 21 where 5 mm/sec was used in order to show the estimates more clearly. The scales for channels 1, 2, 4 and 6, and for channel 7 gain option, are 5 v./div., with 100 volts corresponding to a variable value of 1.0; the scales for channels 3 and 5 are 2 v./div. For the channel 7 variance option the scale is 200 mv./div., and for channel 8 the scale was calibrated at 4 msec/div.

## F. RESULTS OF TESTS

The outputs of the processor were decoded by hand to determine the error rates. Using the estimate of T, a mark was decoded as a dot if its duration was 2T or less and as a dash otherwise. Similarly, a space was decoded as an element-space if its duration was 2T or less, as a letter-space if the duration was between 2T and 4T, and as a word space otherwise.

The following data were obtained for each run:

1. Letter error rate and bit error rate with no processing.
2. Same error rates with filtering only.
3. Same error rates with filtering and smoothing.

A letter error occurs when any transmitted letter is not correctly decoded. Only one error per transmitted letter is counted; for example, if "A" is decoded as "ET", one

letter error has occurred. On the other hand, if "ET" is decoded as "A", two errors have occurred, since neither "E" nor "T" was decoded correctly. Word spaces are counted as one letter per 7 elements. The sample size in each case was approximately 200 letters. A bit error is defined as at least one mark-space error occurring within a transmitted element duration. Again, only one error per element is counted, and the sample size was approximately 200 bits.

The results of this analysis are presented in Tables III through VI. Column 1 lists the signal-to-noise ratio used without a 100 Hz pre-detection filter; column 2 lists the signal-to-noise ratio used with the 100 Hz filter in place. Both the bit and letter error rates for filtering and smoothing are tabulated, with the error rates for the unprocessed output shown for comparison. In each case, the error rates are shown for the typical (random letter) sequence and the $\overline{AR}$ sequence, except in Table VI, where the results are for the $\overline{AR}$ sequence only since no random letter sequence for this case was recorded. Table VII shows typical hand-translated sequences for each processing stage.

These results indicate that the Kalman filter and linear smoother provide a significant decrease in both bit and letter error rates. By using a 100 Hz bandpass pre-detection filter, such processing provides a tolerable 10% letter error rate on a -7 dB SNR signal as opposed to an unacceptable

Channel

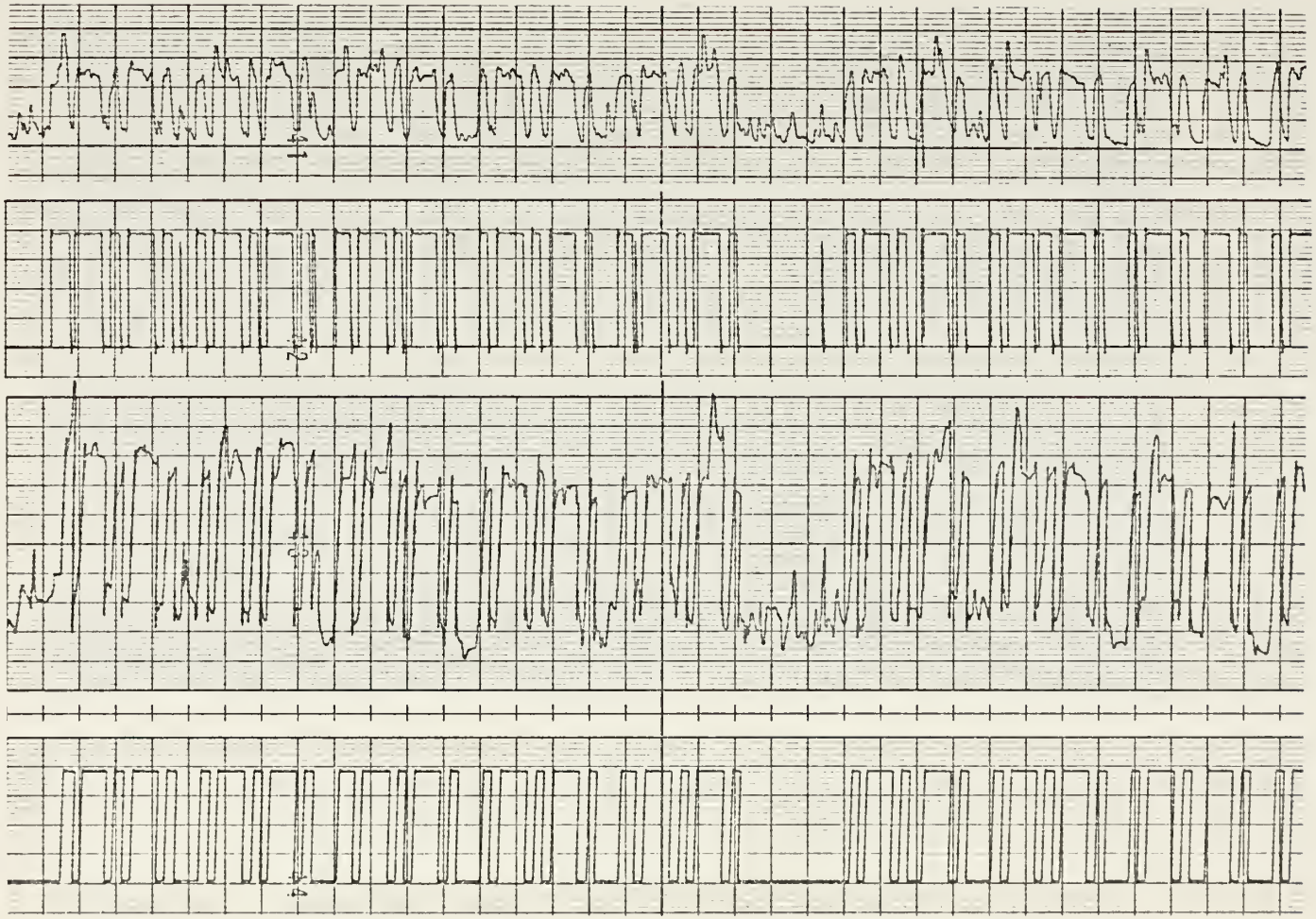



FIGURE 9a. Filter Output for 35 wpm Sequence, 6 dB SNR

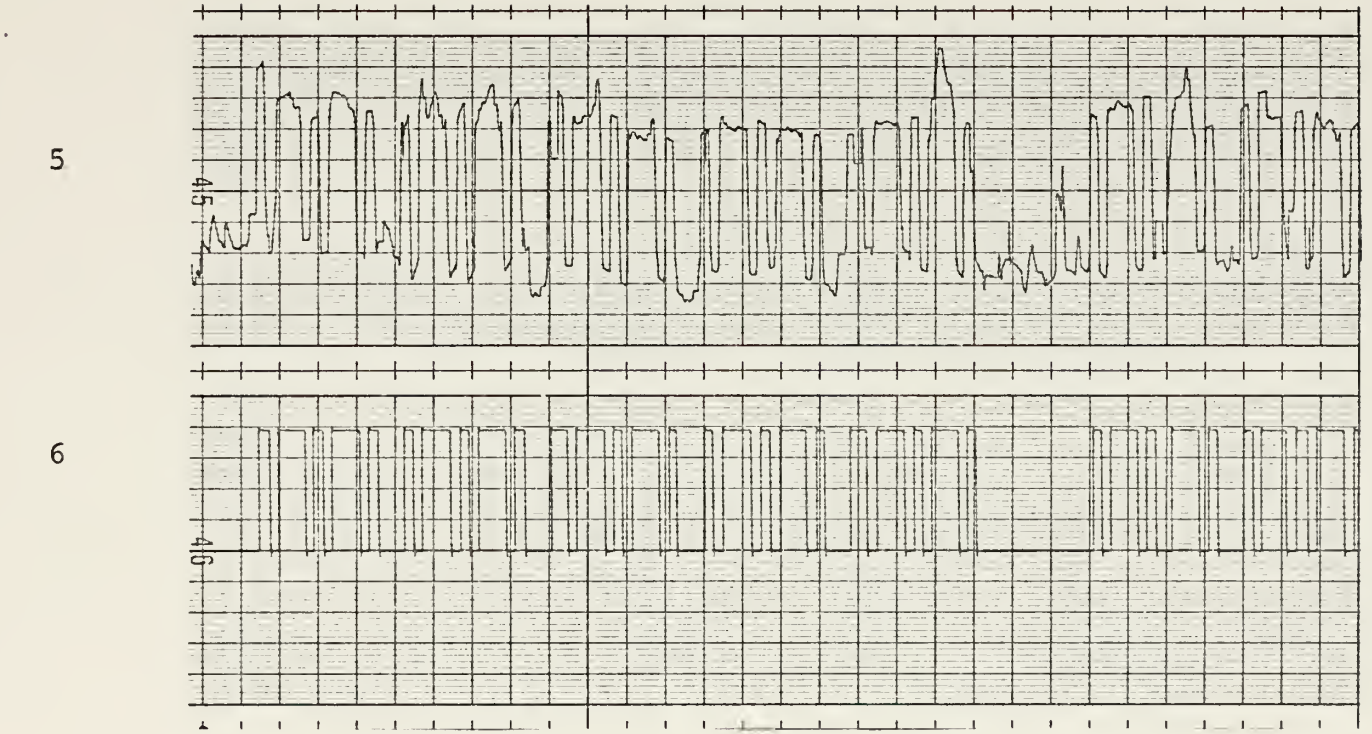



FIGURE 9b. Smoothed Output for 35 wpm Sequence, 6 dB SNR

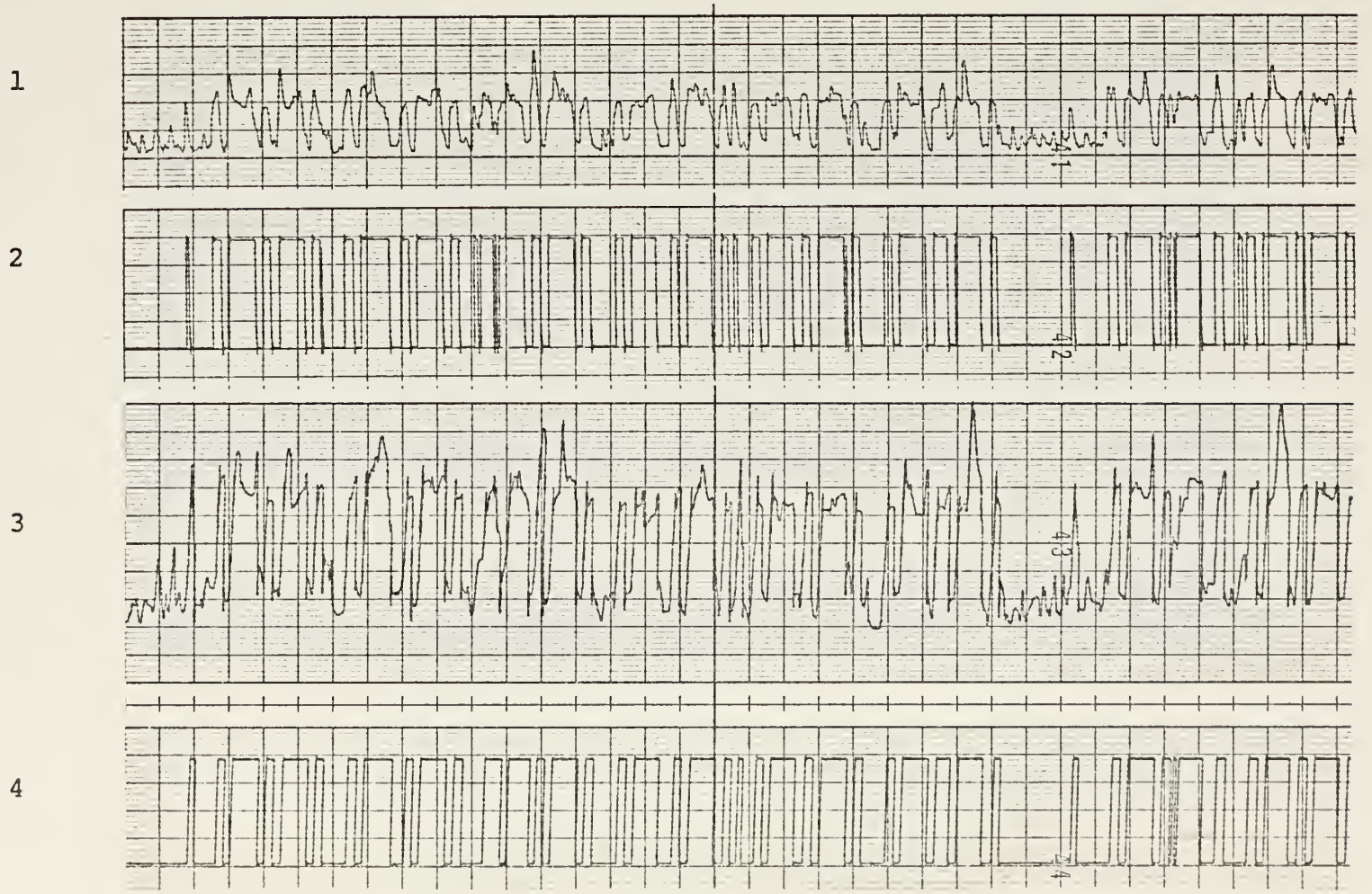



FIGURE 10a. Filter Output for 35 wpm Sequence, 5 dB SNR

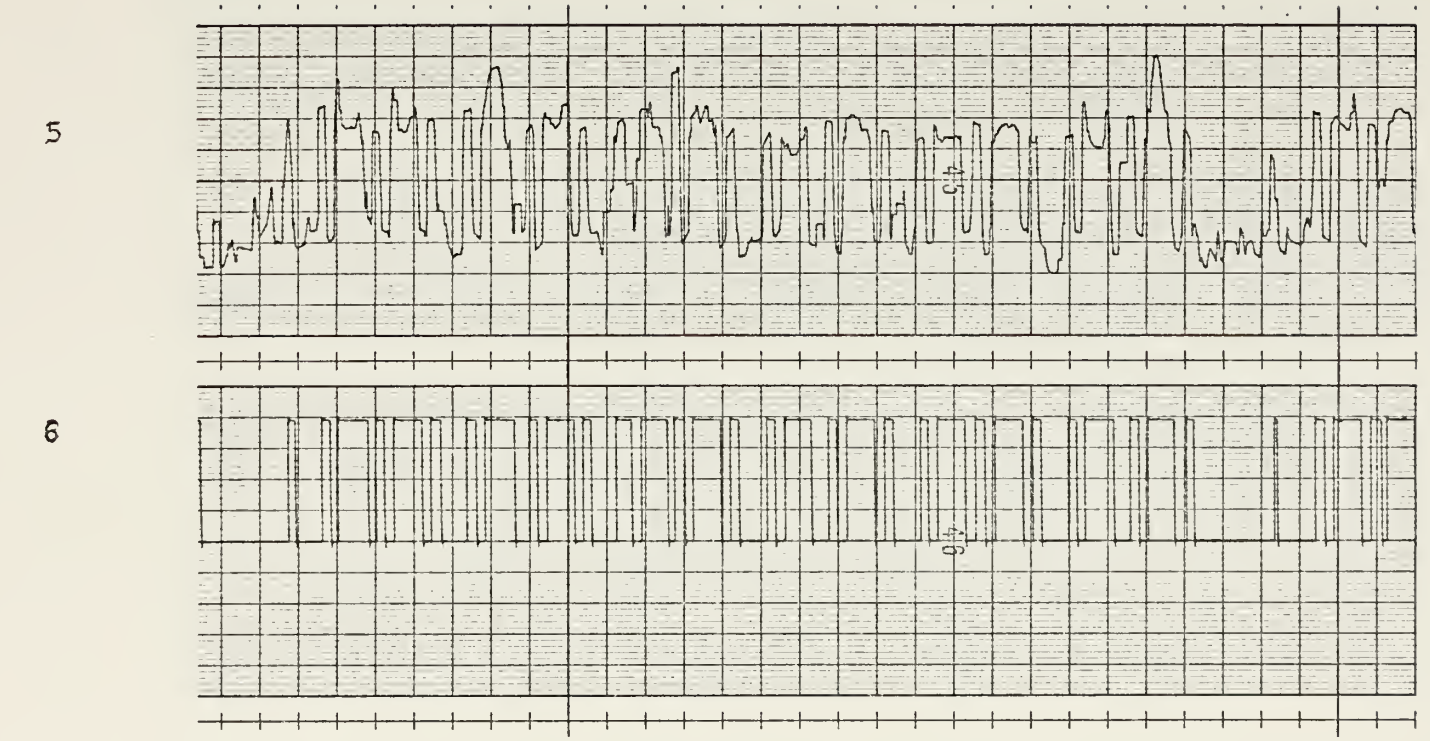



FIGURE 10b. Smoothed Output for 35 wpm Sequence, 5 dB SNR

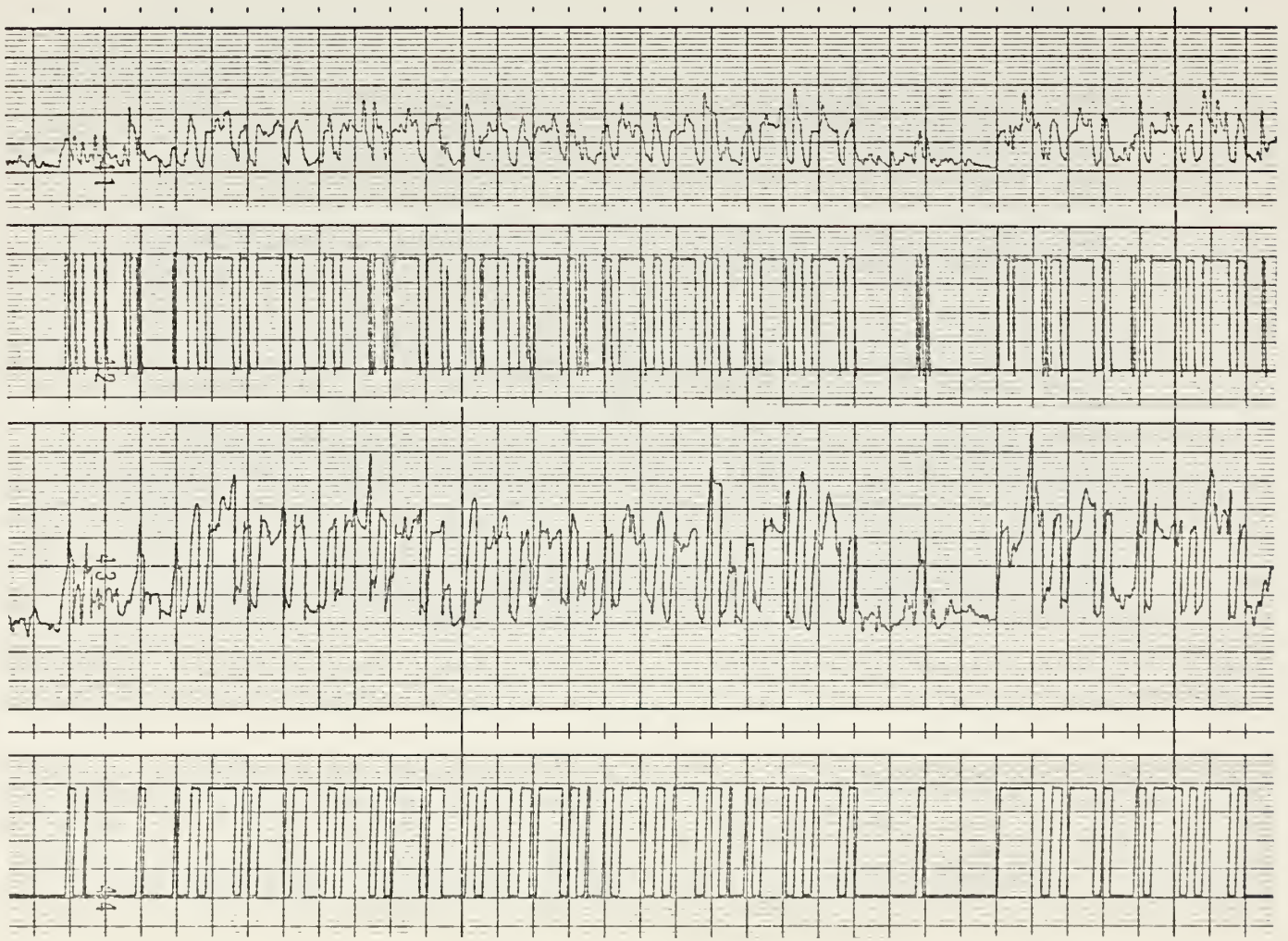



FIGURE 11a. Filter Output for 35 wpm $\overline{AR}$ Sequence, 4 dB SNR

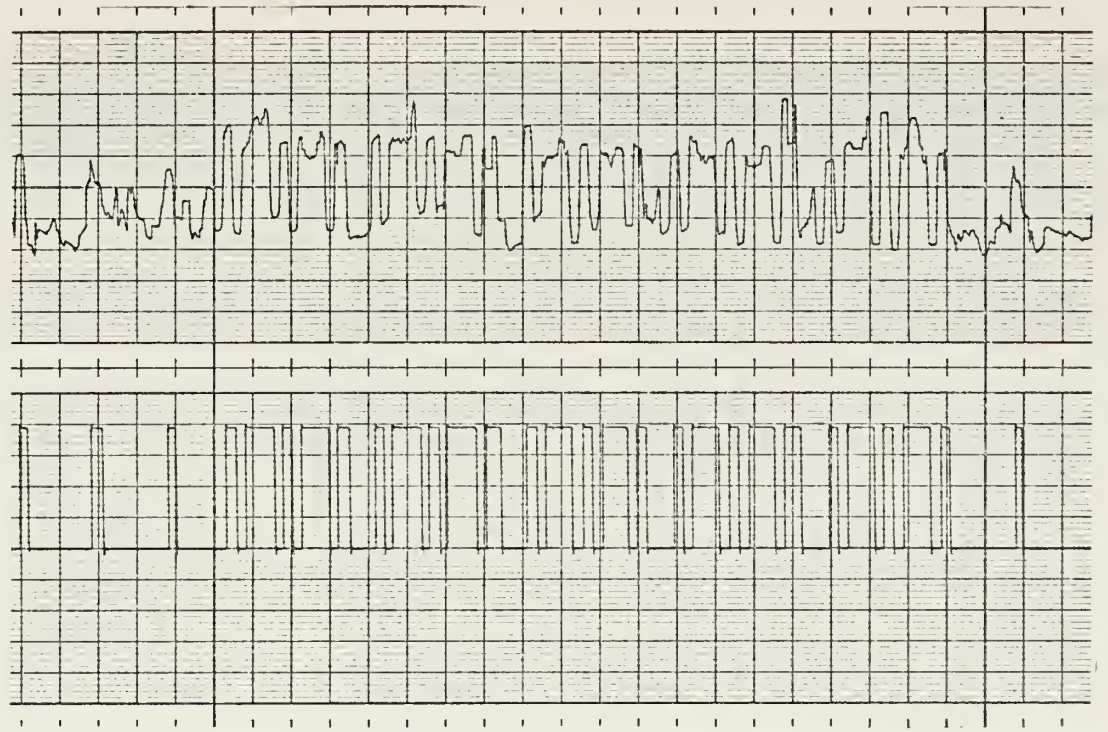



FIGURE 11b. Smoothed Output for 35 wpm $\overline{AR}$ Sequence, 4 dB SNR

Channel

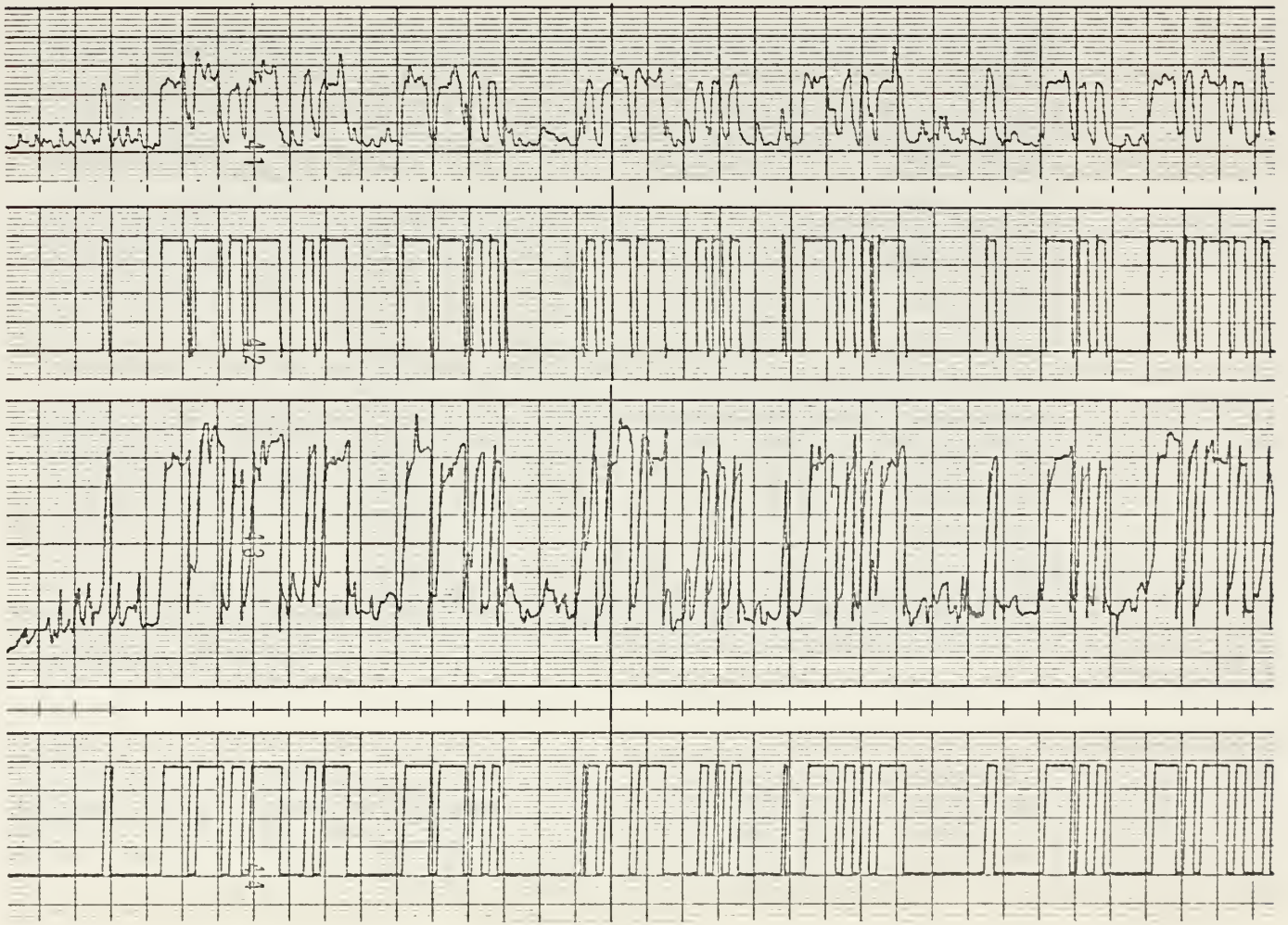



FIGURE 12a. Filter Output for 35 wpm Typical Sequence, 6 dB SNR


FIGURE 12b. Smoothed Output for 35 wpm Typical Sequence, 6 dB SNR

Channel

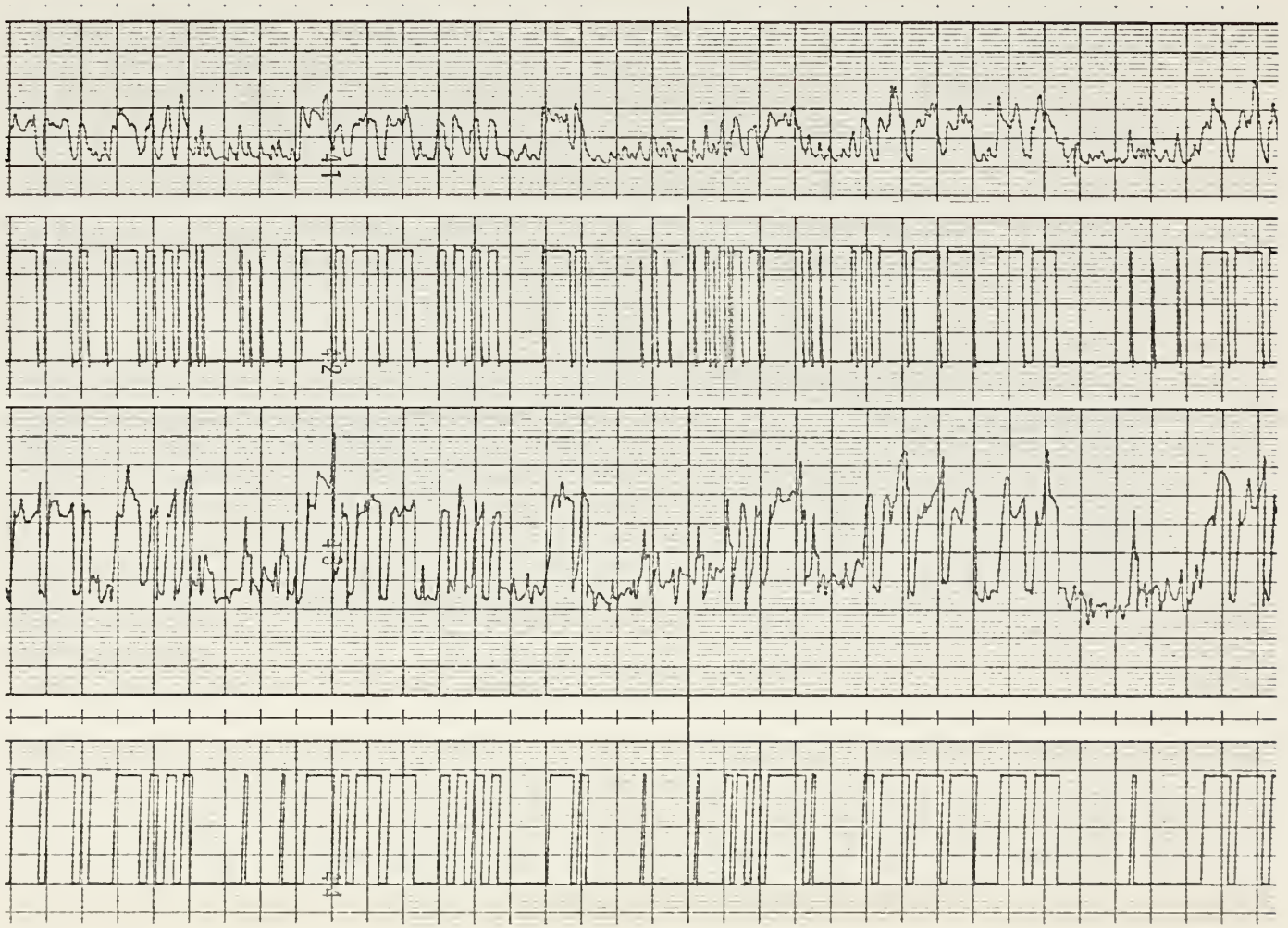



FIGURE 13a. Filter Output for 35 wpm Typical Sequence, 4 dB SNR

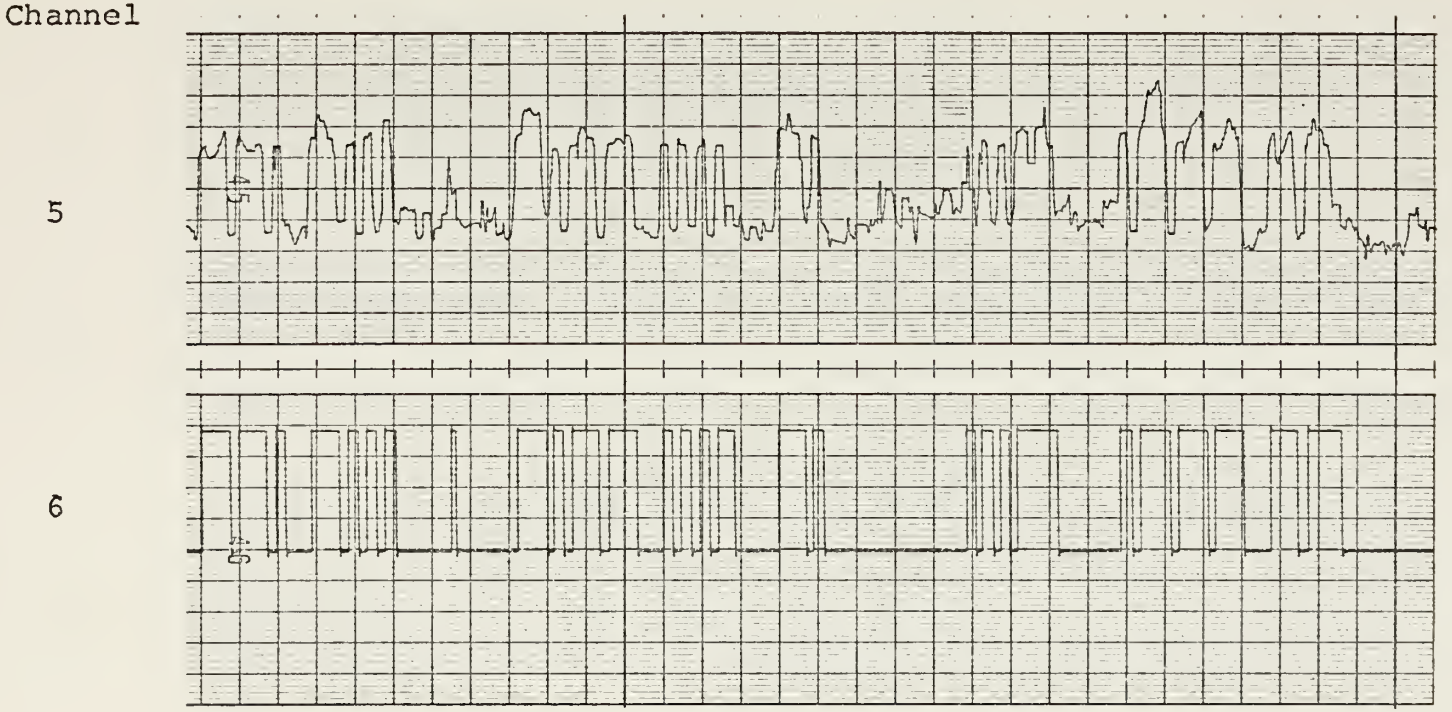



FIGURE 13b. Smoothed Output for 35 wpm Typical Sequence, 4 dB SNR

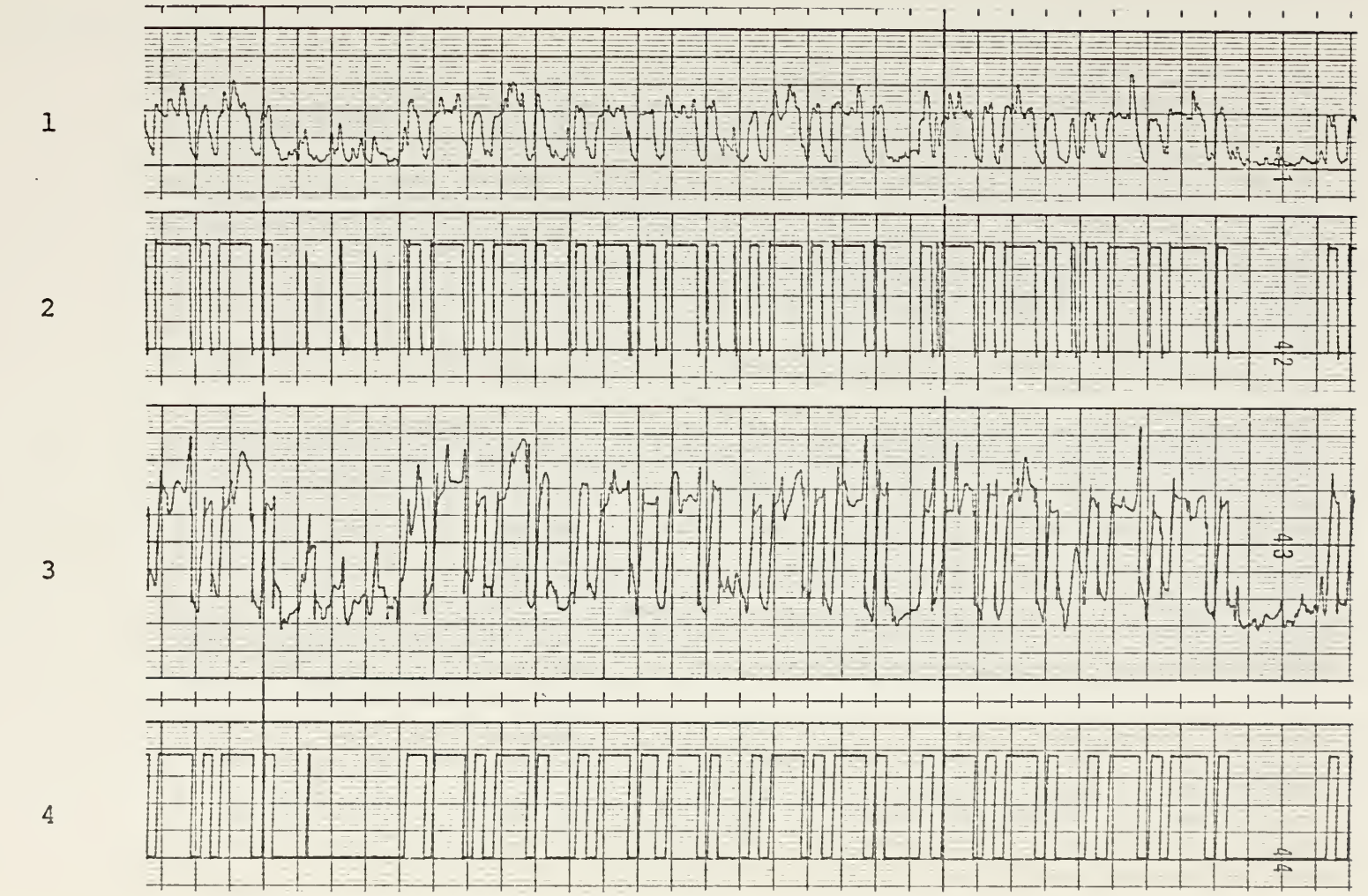



FIGURE 14a. Filter Output for 30 wpm $\overline{AR}$ Sequence, 5 dB SNR

Channel

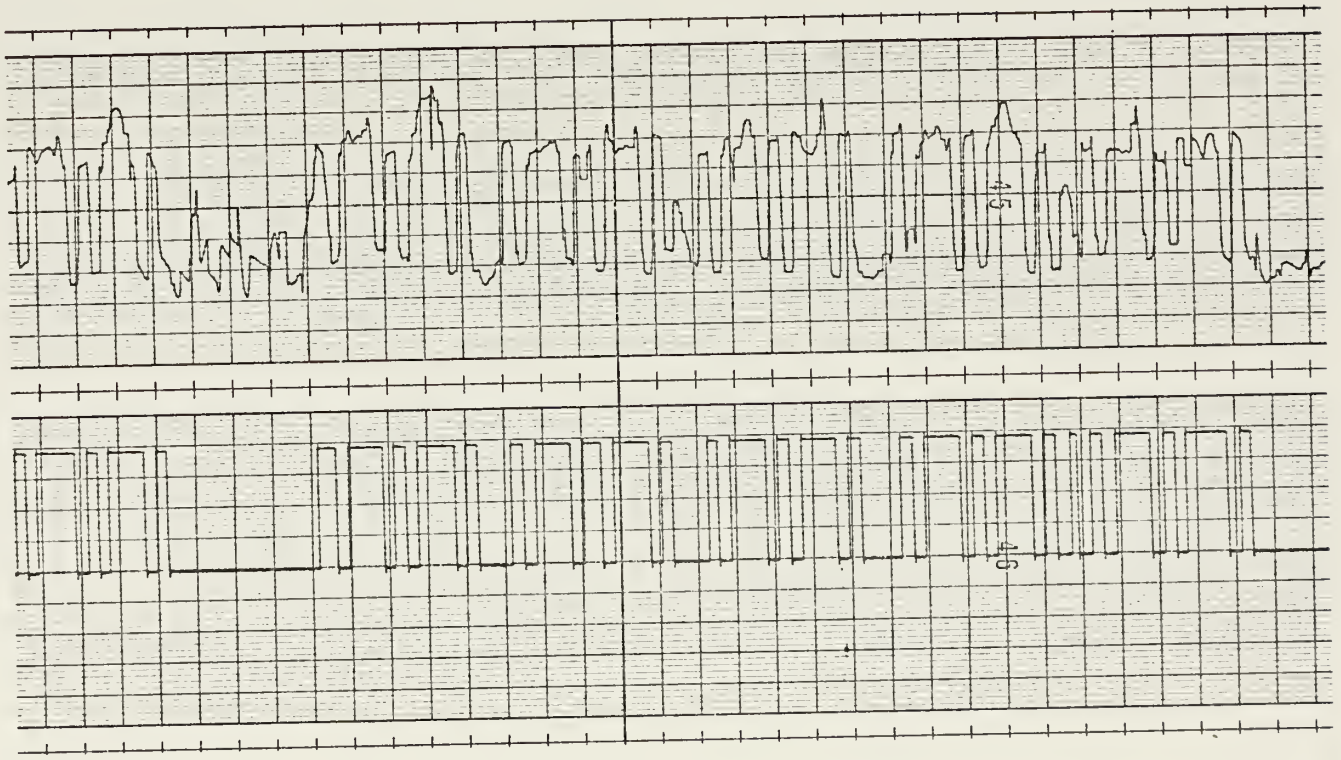



FIGURE 14b. Smoothed Output for 30 wpm $\overline{AR}$ Sequence, 5 dB SNR

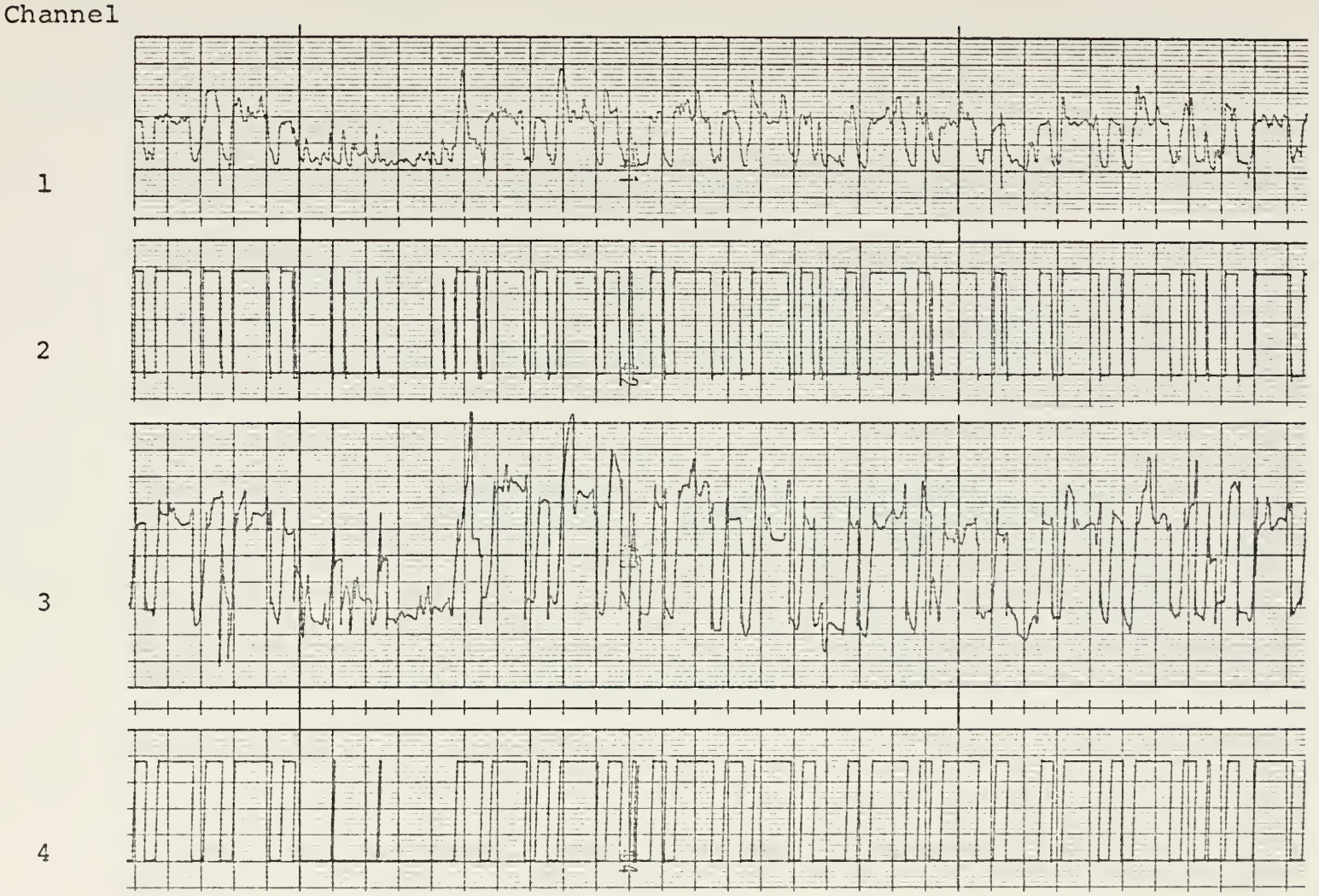



FIGURE 15a. Filter Output for 25 wpm $\overline{AR}$ Sequence, 5 dB SNR

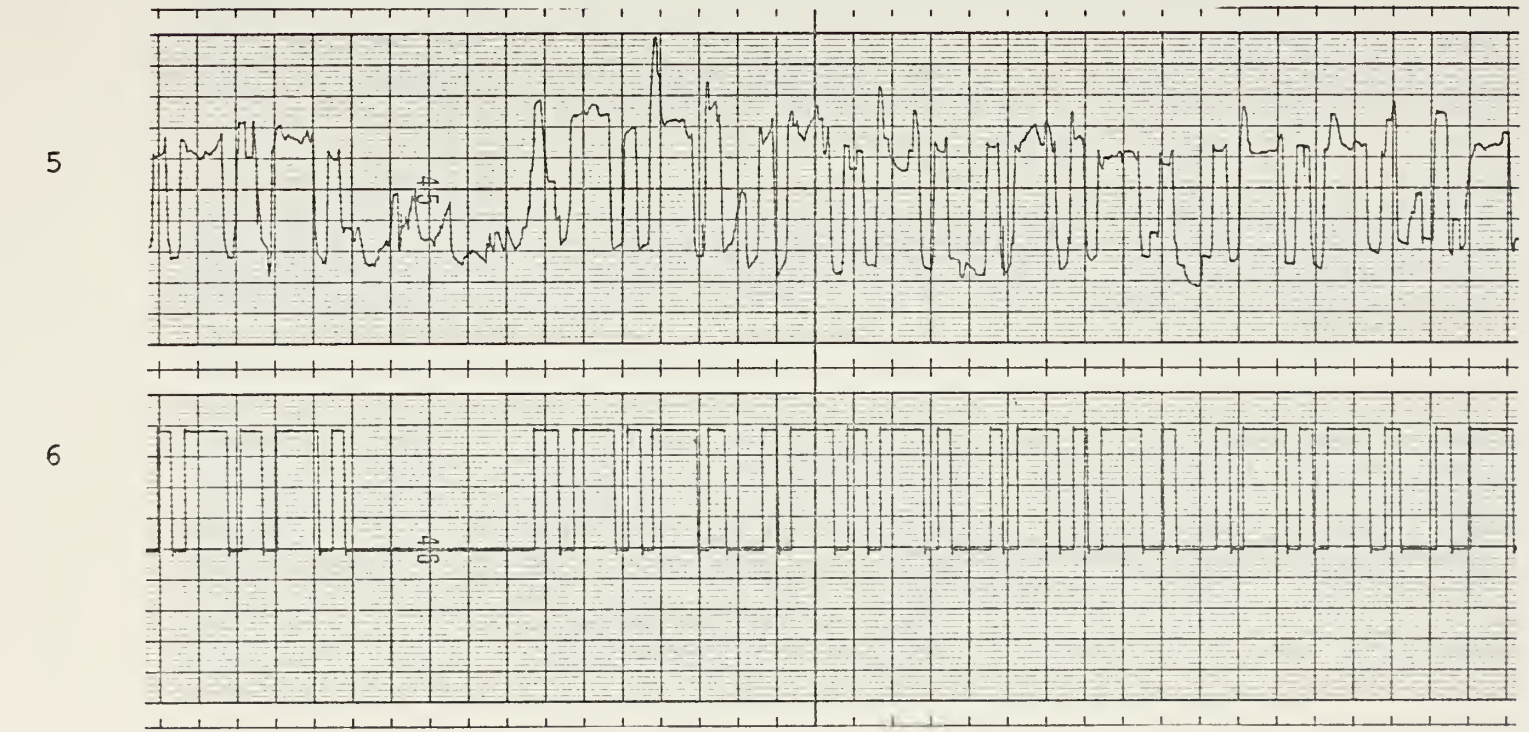



FIGURE 15b. Smoothed Output for 25 wpm $\overline{AR}$ Sequence, 5 dB SNR

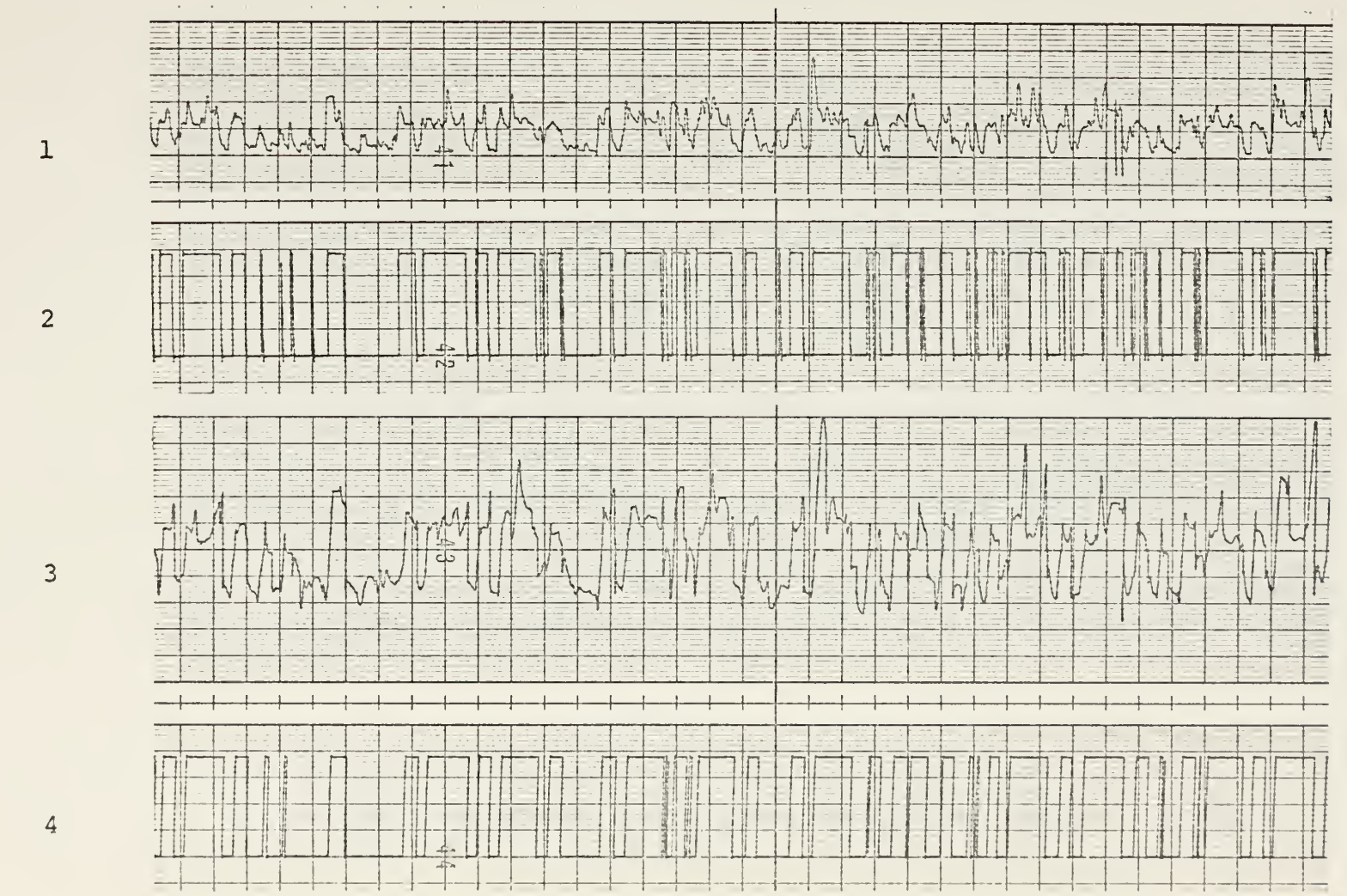



FIGURE 16a. Filter Output for 25 wpm $\overline{AR}$ Sequence, 3 dB SNR


FIGURE 16b. Smoothed Output for 25 wpm $\overline{AR}$ Sequence, 3 dB SNR

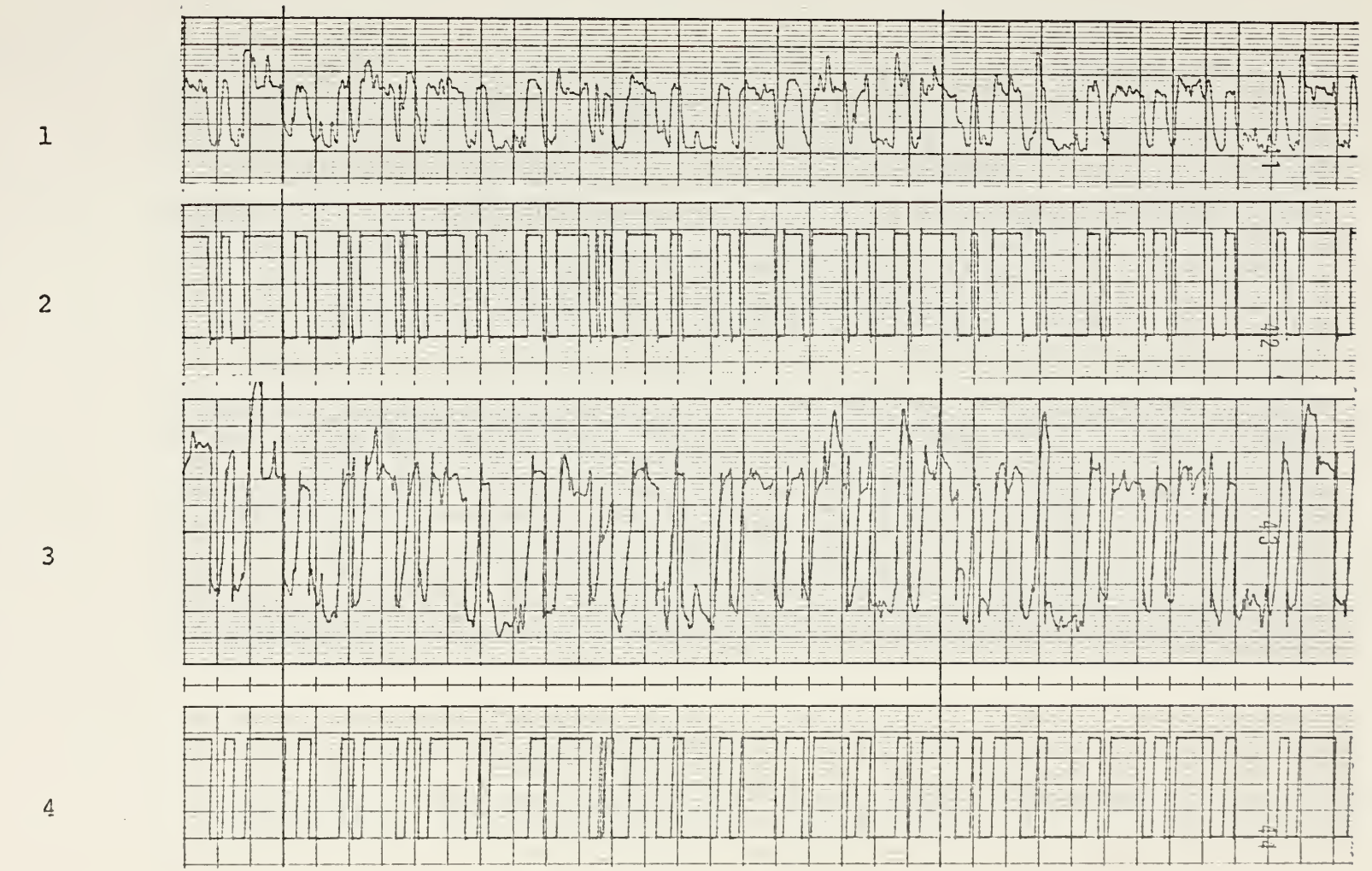



FIGURE 17a. Filter Output for 25 wpm Sloppy Code $\overline{AR}$ Sequence, 6 dB SNR

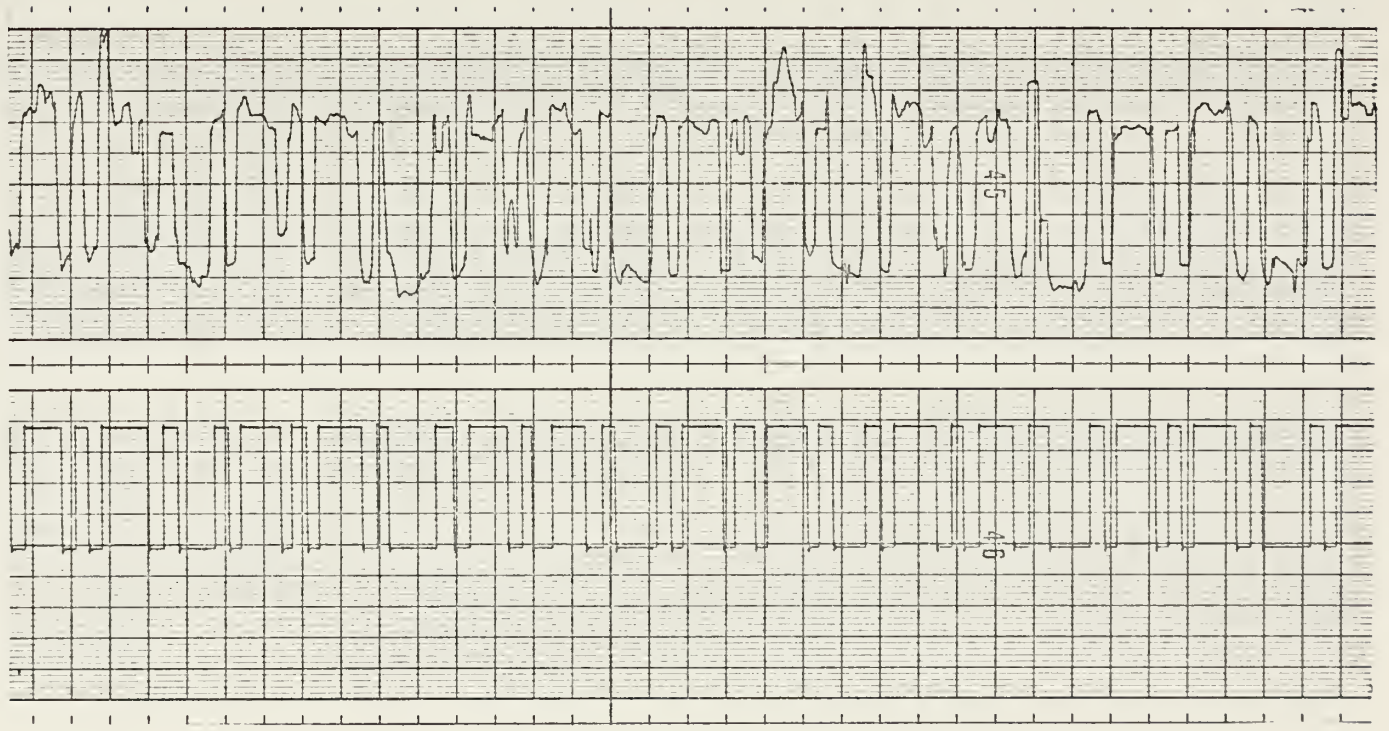



FIGURE 17b. Smoothed Output for 25 wpm Sloppy Code $\overline{AR}$ Sequence, 6 dB SNR

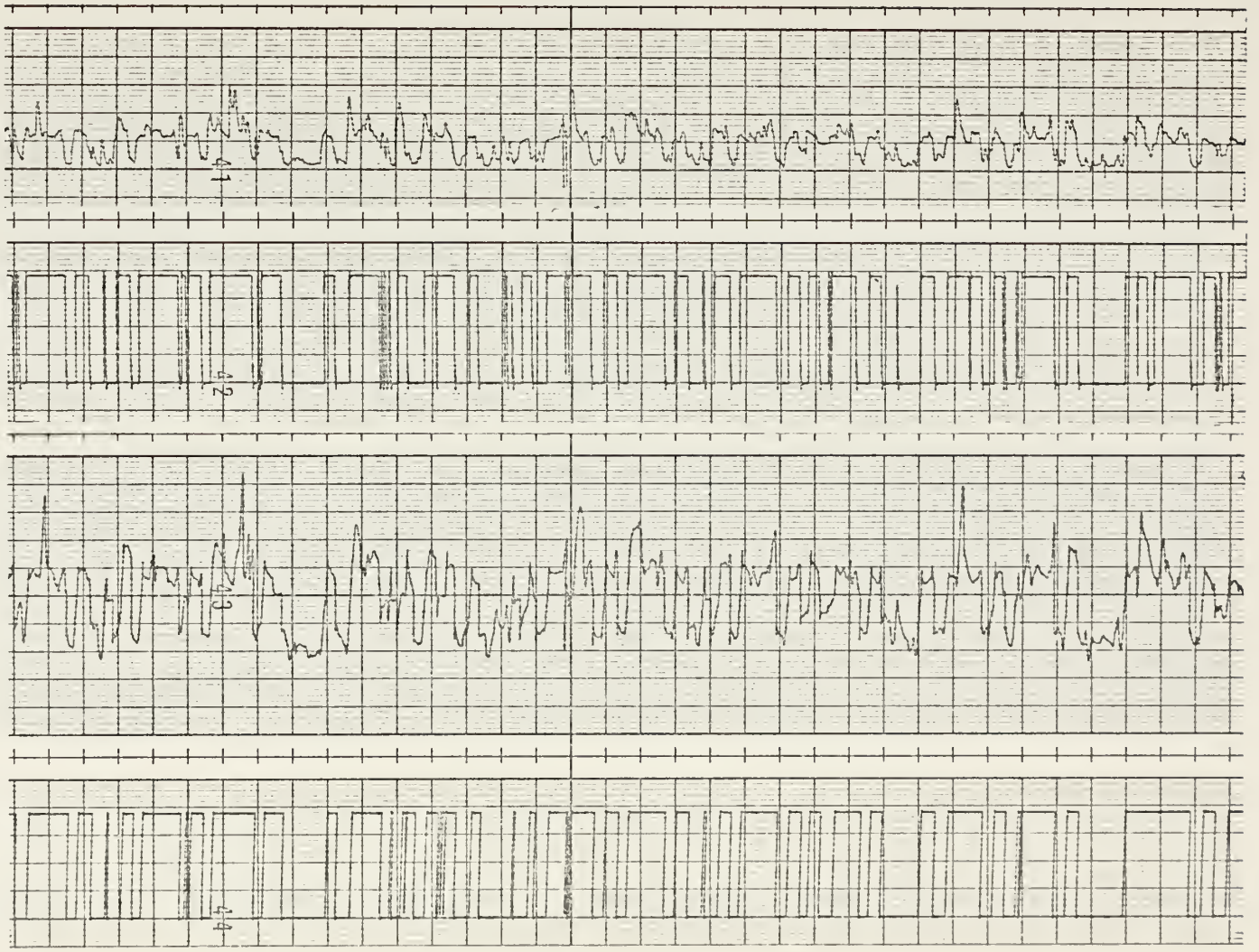



FIGURE 18a. Filter Output for 25 wpm Sloppy Code $\overline{AR}$ Sequence, 3 dB SNR

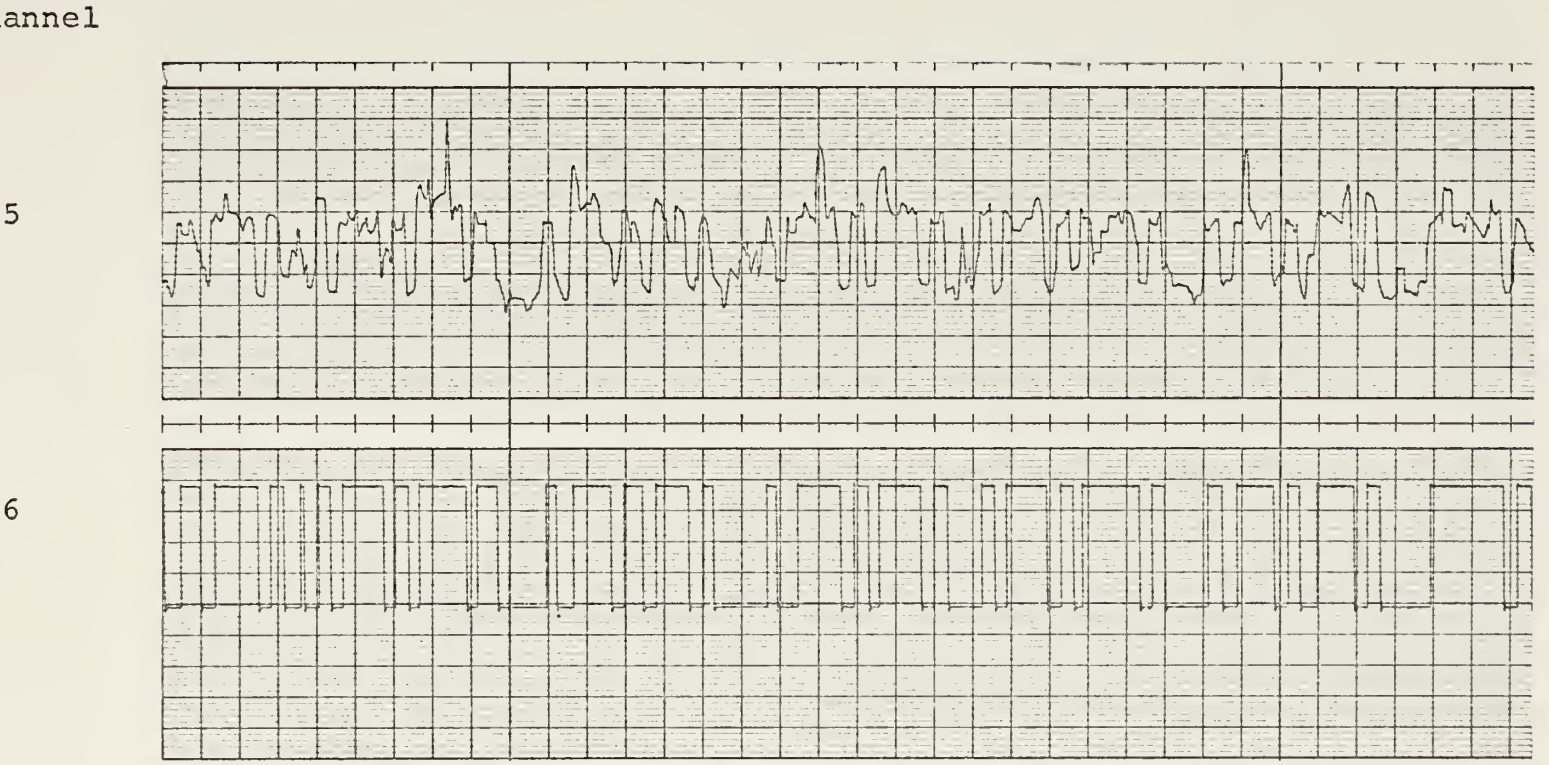



FIGURE 18b. Smoothed Output for 25 wpm Sloppy Code $\overline{\text{AR}}$ Sequence, 3 dB SNR

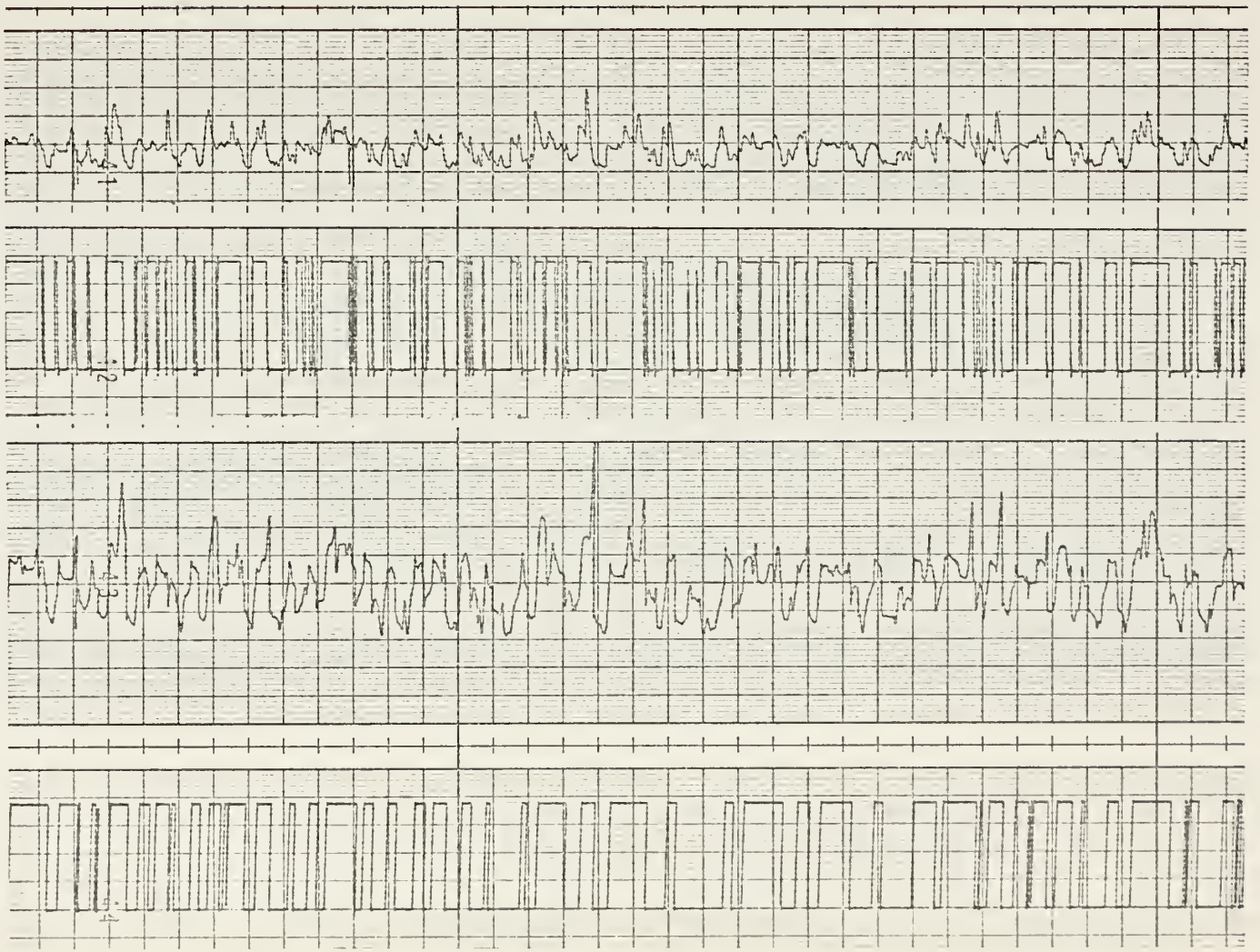



FIGURE 19a. Filter Output for 25 wpm Sloppy Code $\overline{AR}$ Sequence, 1 dB SNR

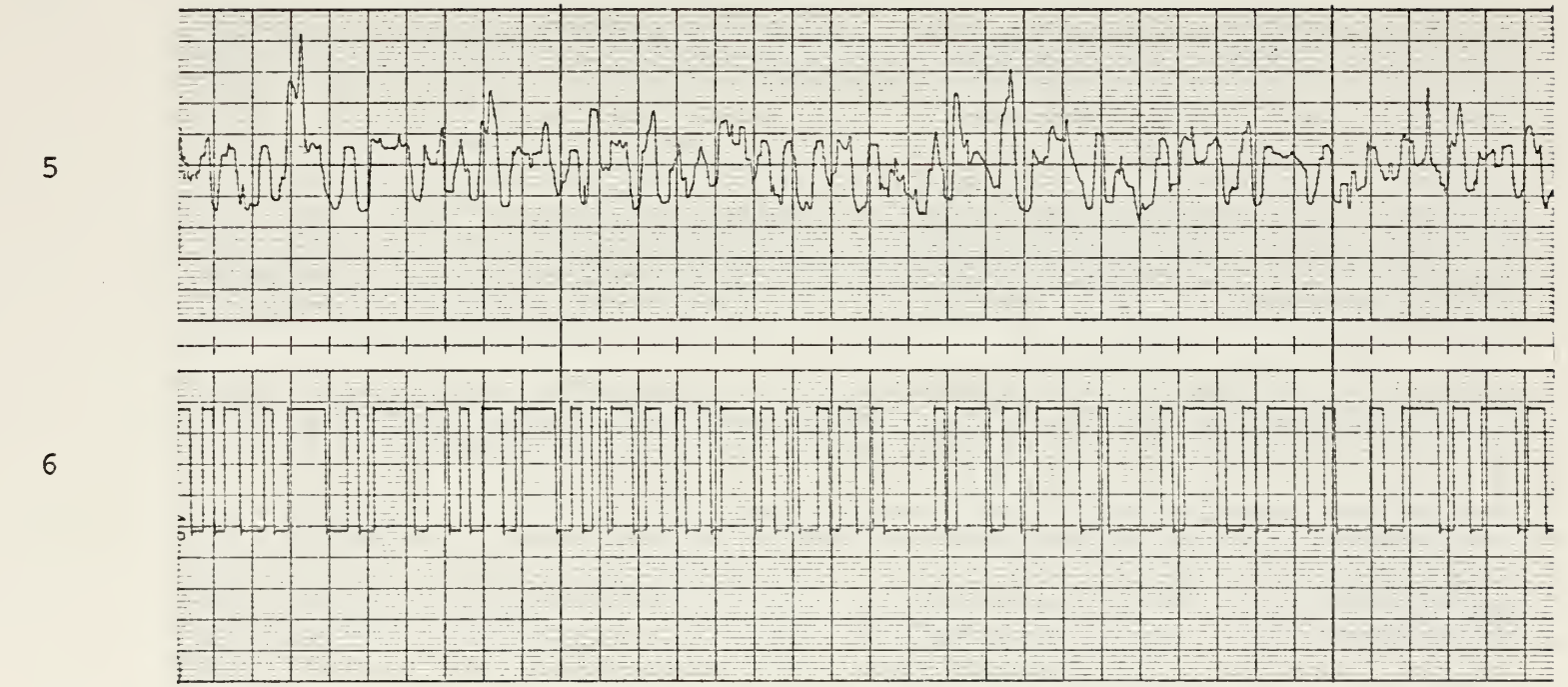



FIGURE 19b. Smoothed Output for 25 wpm Sloppy Code $\overline{AR}$ Sequence, 1 dB SNR

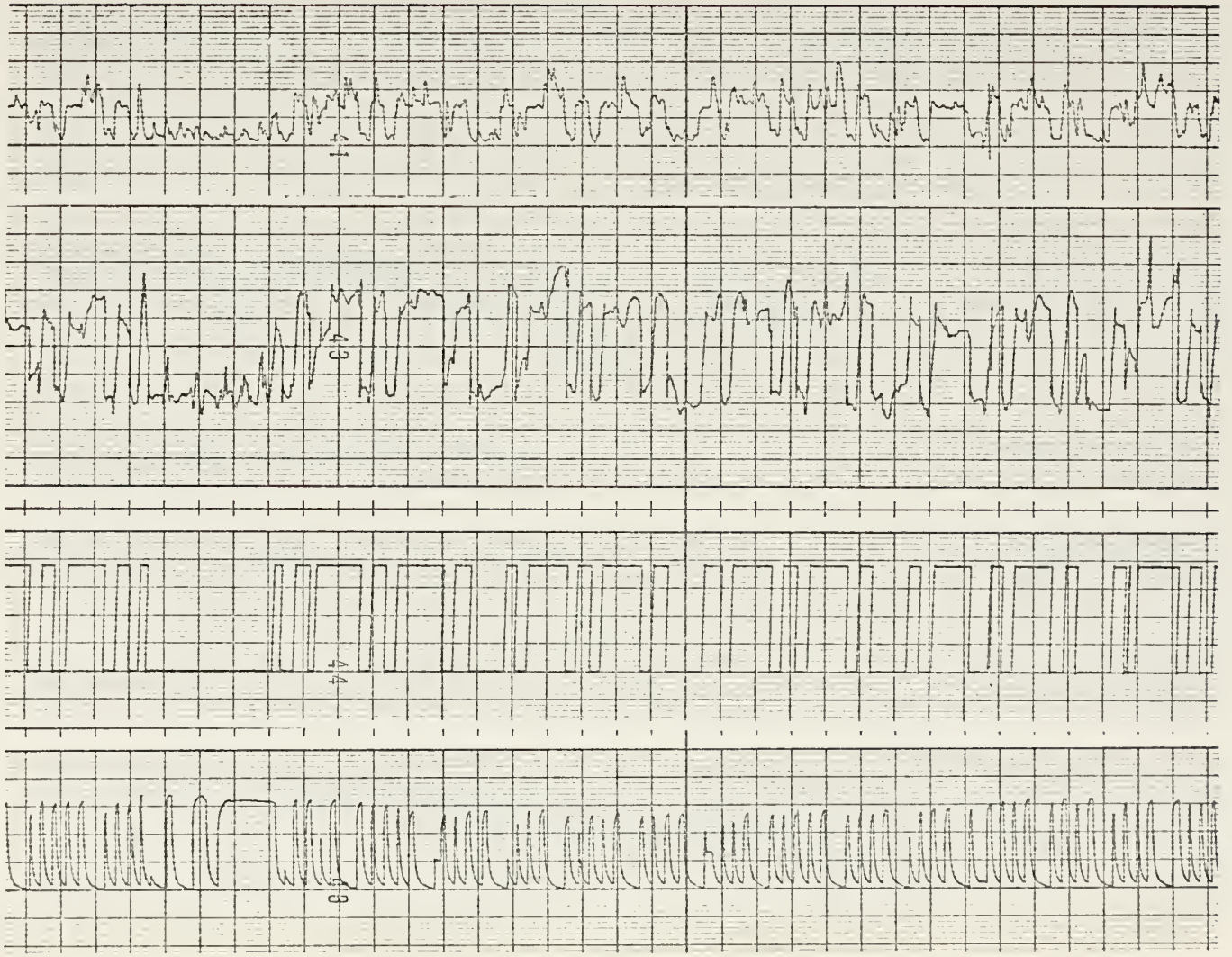



FIGURE 20. Filter Gain (Channel 7) 4 dB SNR

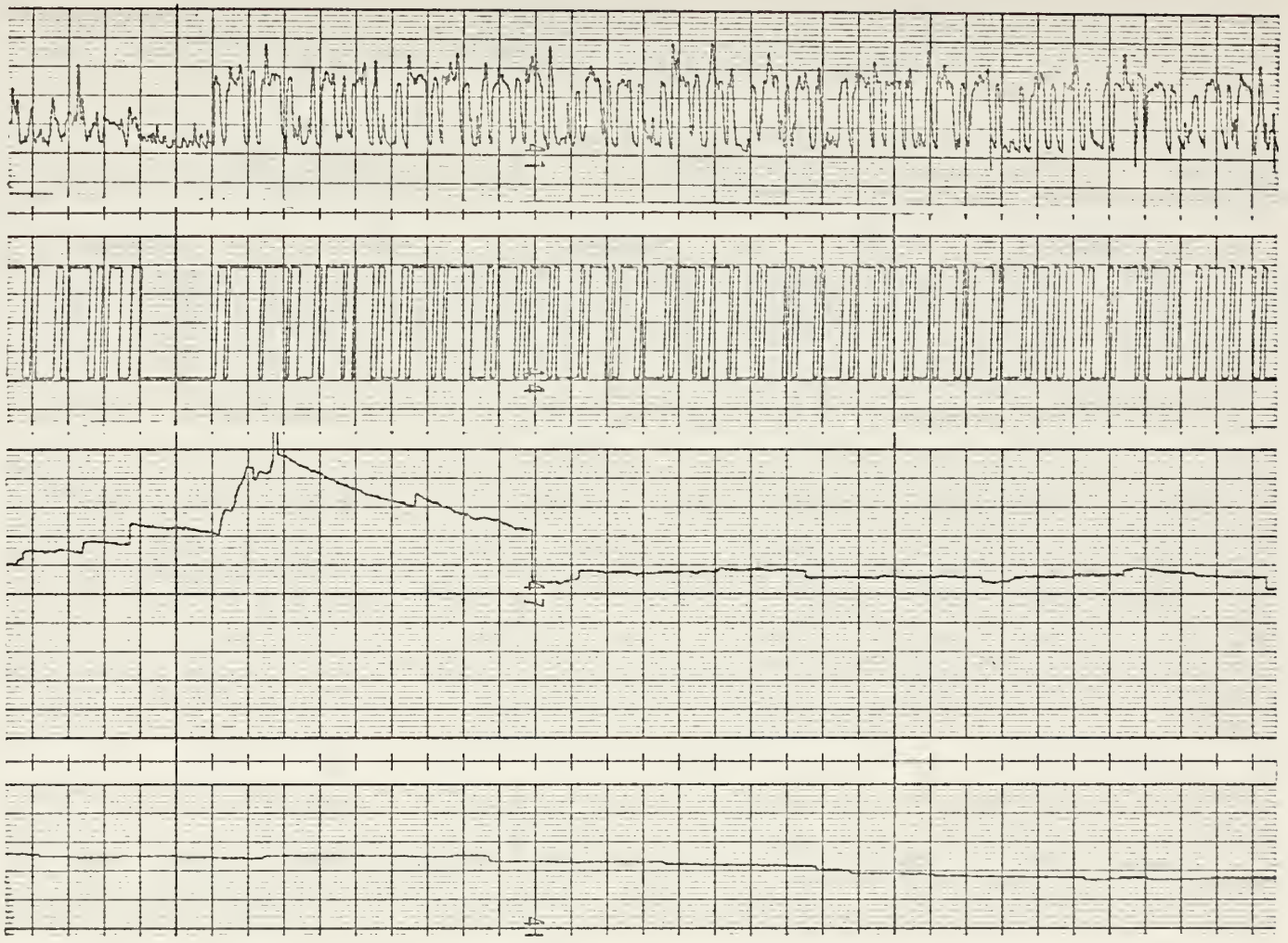



FIGURE 21. Noise Variance and Estimate of T

TABLE III

ERROR RATES FOR CODE SPEED OF 35 wpm

| SNR (2 kHz) | | NO PROCESSING | | | | FILTERED | | | | SMOOTHED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| no filter (db) | 100 Hz filter (db) | Typical | | $\overline{AR}$ | | Typical | | $\overline{AR}$ | | Typical | | $\overline{AR}$ | |
| | | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) |
| 6 | -7 | 3.4 | 32 | 3.5 | 34 | 0.81 | 9.0 | 1.1 | 12 | 0.70 | 8.0 | 0.83 | 9.0 |
| 5 | -8 | 7.4 | 46 | 8.4 | 54 | 2.2 | 25 | 3.2 | 28 | 1.2 | 15 | 3.0 | 27 |
| 4 | -9 | 14 | 63 | 14 | 62 | 5.0 | 36 | 7.0 | 47 | 4.5 | 34 | 6.7 | 49 |
| 3 | -10 | 23 | 85 | 28 | 89 | 20 | 84 | 19 | 86 | 30 | 92 | 35 | 96 |
| 2 | -11 | 34 | 93 | 37 | 91 | 35 | 99 | 30 | 99 | 35 | 100 | 35 | 100 |
| 1 | -12 | ~50 | 100 | ~50 | 100 | ~50 | 100 | ~50 | 100 | ~50 | 100 | ~50 | 100 |

TABLE IV

ERROR RATES FOR CODE SPEED OF 30 wpm

| SNR (2 kHz) | | NO PROCESSING | | | | FILTERED | | | | SMOOTHED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| no filter (db) | 100 Hz filter (db) | Typical | | $\overline{AR}$ | | Typical | | $\overline{AR}$ | | Typical | | $\overline{AR}$ | |
| | | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) |
| 6 | -7 | 3.3 | 32 | 3.2 | 32 | 1.9 | 19 | 2.9 | 30 | .82 | 9.0 | 0.92 | 10 |
| 5 | -8 | 7.6 | 53 | 8.5 | 54 | 2.0 | 23 | 3.1 | 28 | 1.0 | 13 | 1.7 | 19 |
| 4 | -9 | 15 | 65 | 17 | 64 | 2.9 | 30 | 5.3 | 41 | 2.4 | 27 | 3.8 | 33 |
| 3 | -10 | 23 | 89 | 30 | 91 | 5.1 | 49 | 7.6 | 54 | 4.5 | 36 | 4.7 | 41 |
| 2 | -11 | 33 | 95 | 36 | 98 | 11.1 | 60 | 9.8 | 63 | 8.2 | 59 | 8.5 | 61 |
| 1 | -12 | ~50 | 100 | ~50 | 100 | ~50 | 100 | 37 | 100 | 35 | 95 | 35 | 95 |

TABLE V

ERROR RATES FOR CODE SPEED OF 25 wpm

| SNR (2kHz) | | NO PROCESSING | | | | FILTERED | | | | SMOOTHED | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No Filter | 100 Hz Filter | Typical | | $\overline{AR}$ | | Typical | | $\overline{AR}$ | | Typical | | $\overline{AR}$ | |
| (db) | (db) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) |
| 6 | -7 | 3.3 | 31 | 3.2 | 32 | 1.3 | 15 | 1.9 | 22 | 0.70 | 8 | 0.77 | 10 |
| 5 | -8 | 7.7 | 50 | 7.5 | 48 | 1.5 | 17 | 1.8 | 20 | 1.1 | 12 | 1.3 | 17 |
| 4 | -9 | 13 | 61 | 14 | 62 | 2.1 | 21 | 2.4 | 28 | 1.2 | 15 | 1.7 | 18 |
| 3 | -10 | 22 | 85 | 25 | 84 | 2.9 | 30 | 5.1 | 46 | 2.9 | 24 | 4.5 | 36 |
| 2 | -11 | 34 | 93 | 33 | 94 | 7.4 | 50 | 9.2 | 57 | 6.7 | 51 | 8.6 | 57 |
| 1 | -12 | ~50 | 100 | ~50 | 100 | 23 | 85 | 31 | 95 | 21 | 93 | 30 | 95 |

TABLE VI

ERROR RATES FOR SLOPPY CODE

| SNR (2 kHz) | | 25 wpm | | | | | | 30 wpm | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | No Processing | | Filtered | | Smoothed | | No Processing | | Filtered | | Smoothed | |
| No Filter (db) | 100 Hz Filter (db) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) | Bit (%) | Ltr (%) |
| 6 | -7 | 2.2 | 25 | 1.8 | 19 | .39 | 12 | 1.7 | 19 | 1.2 | 14 | 0.60 | 7.8 |
| 5 | -8 | 8.1 | 50 | 1.8 | 19 | 1.1 | 9.3 | 6.4 | 43 | 1.9 | 22 | 1.2 | 14 |
| 4 | -9 | 17 | 76 | 2.7 | 31 | 1.3 | 13 | 14 | 56 | 3.2 | 32 | 2.8 | 33 |
| 3 | -10 | 29 | 91 | 3.8 | 33 | 2.7 | 26 | 19 | 62 | 3.6 | 29 | 2.4 | 22 |
| 2 | -11 | 35 | 96 | 7.4 | 49 | 7.1 | 53 | 35 | 94 | 7.8 | 55 | 7.5 | 58 |
| 1 | -12 | 44 | 100 | 10 | 52 | 10 | 59 | ~50 | 100 | ~50 | 100 | ~50 | 100 |
| | | DOT VARIATION 30-50 msec<br>ELEMENT-SPACE VARIATION 60-40 msec<br>DASH VARIATION 120-150 msec | | | | | | DOT VARIATION 25-40 msec<br>ELEMENT-SPACE VARIATION 50-30 msec<br>DASH VARIATION 100-120 msec | | | | | |

TABLE VII

TYPICAL TRANSLATED SEQUENCES

| | |
|---|---|
| SENT | QA Z WS X E DC R F V T GB YHN UJM |
| UNPROCESSED | ?? ? E WH XEIED? E FFV T GG E YHN FJM |
| FILTERED | ?? ? E WH X IEDC E RFV T GG YHN UJM |
| SMOOTHED | ?? ? WH X E DC E RFV T GB YHN UJM |

| | |
|---|---|
| SENT | QAZ WS X E DC RFV T GB YHN UJM |
| UNPROCESSED | IEQAZ PSEX I DCEFF?EA GBEYHNEU?? |
| FILTERED | IEQAZ ?? X I DC FFVIT GB YHN UJM |
| SMOOTHED | EEQAZ ?? X T DC RF? T GB YHN UJM |

| | |
|---|---|
| SENT | QAZ WSX E E DC RFV T GB YHN UJM |
| UNPROCESSED | QUAD ??XEEI I DC E LPVET PBEYHNE?JME |
| FILTERED | QMZ WHXEEI I DY E RFV T GB YUN F?ME |
| SMOOTHED | QAZ WHX I I DC RFV T GB QUNEUJM |

| | |
|---|---|
| SENT | QAZ WS X E DC RFV T GB YHN U |
| UNPROCESSED | Q?EZ E LS X EIH ?? LFVIA GB E YH?EV |
| FILTERED | QAZ E WS X EEE DC E RFVET GB YHN U |
| SMOOTHED | QAZ WS X E D? E RFV T GB YHN U |

SUMMARY:

| | | Error Rate |
|---|---|---|
| total letters sent | 125 | - |
| unprocessed errors | 58 | 46% |
| filtered errors | 31 | 25% |
| smoothed errors | 19 | 15% |

(question mark indicates untranslatable sequence)

BW = 2 kHz SNR = 5 dB speed = 30 wpm perfect code

32-34% error rate without processing. Smoothing contributes considerably to a reduced error rate down to about -9 dB where its effectiveness over filtering alone begins to fall off. For the high speed 35 wpm signal, runaway of Q estimation occurs at approximately -9 dB, while for the lower speeds runaway never really develops until the SNR reaches -12 dB, since filtering always provides an improvement in error rate even at these low SNR's.

It was noted that a majority of the errors in the filtered and smoothed output result from insertions of isolated dots in the letter-space and particularly the word-space separations. A possible remedy to this situation is to incorporate the Markov structure of the code in the estimation algorithm for Q, although an increased susceptibility to runaway may limit its effectiveness.

Since the bit error rate must be on the order of 1% or less in order to yield a tolerable 10% letter error rate, it was felt that bit error rates of up to approximately 3% could possibly be reduced to an acceptable level by use of soft-decision Viterbi decoding following the smoothed output. If such a reduction were possible, then acceptable error rates could be obtained for SNR's down to -9 dB.

Additionally, pre-detection Kalman filtering and parameter estimation would yield a theoretical gain of 3 dB or more over non-coherent demodulation. With these improvements in the processor, then, the output letter error rate of 10%

could be achieved on signals whose SNR is on the order of -12 dB in a 2 kHz bandwidth and the originally stated design objective would be met.

Both of these possible improvements were implemented separately to determine the effectiveness of each one. The Viterbi decoder algorithm used the smoothed output of the post-detection Kalman filter as input. A separate program was written to implement a pre-detection Kalman filter.

## VI. VITERBI DECODER

The Viterbi algorithm [7],[8], as originally formulated, is a maximum likelihood (ML) sequence estimation decoding algorithm for convolutional codes. It has found application in other areas [9], however, and its use has been extended to maximum a posteriori (MAP) estimation. It is the MAP estimation use which is of importance here.

### A. MAP ESTIMATION

In order to transform the smoothed output of the processor into characters of the morse code, certain decision criteria must be formulated. The easiest and most obvious way to accomplish decoding is to threshold the smoothed output at its mean value and determine the identity of marks and spaces on the basis of the measured duration of each received character. Such a scheme, however, fails to utilize two sources of information which are inherent in the smoothed output: 1) Thresholding discards all information present in the actual smoothed amplitude estimate, and 2) the Markov nature of the character transitions is not utilized.

The decoding problem, then, is to take advantage of this additional information to determine the most probable sequence of morse characters. The thresholded output may be used to make tentative decisions to obtain a specific sequence of character outputs $z_o,\ldots,z_n$, where a particular

$z_i$ is either a dot, dash, element-space, or letter-space. It is the objective, then, to find a specific sequence of transmitted characters $x_o,\ldots,x_n$ which maximizes the probability that $x_o,\ldots,x_n$ was transmitted, given that $z_o,\ldots,z_n$ was received. Formally, it is desired to find the sequence $\underline{x}_j = x_o,\ldots,x_n$ which gives

$$P(\underline{x}=\underline{x}_j|\underline{z}=\underline{z}_j) = \text{maximum}$$

where $\underline{z}_j$ is the specific received sequence, and all possible sequences of transmitted characters, $\underline{x}$, are used to determine which specific sequence yields the maximum value. The sequence $\underline{x}_j$ which yields the maximum is then the maximum likelihood estimate of the transmitted sequence if it is assumed that the transmitted characters are equiprobable; it is the maximum a posteriori (or minimum error) estimate if the actual probability of transmission of each character is utilized [10].

In general, it would be necessary to compute and compare the probability, $P(\underline{x}=\underline{x}_\ell|\underline{z}=\underline{z}_j)$, for all possible sequences $\underline{x}_\ell$. However, if it is assumed that the morse code is a Markov source, then the problem of finding the MAP estimate reduces to a problem of maximizing a sum and the Viterbi algorithm may be used.

In the following development, a shorthand probability notation is used to facilitate writing of probability statements. The statement $P(x_k|z_k)$ is used to mean

$P(x_k=a_i|z_k=a_j)$ where the $a_i, a_j$ are the characters of the code, i.e., dot, dash, element-space, letter-space. No confusion should result, since in all cases the notation P(u) is intended to mean the probability that u is equal to some specific value.

B. SOURCE MODEL

A third-order Markov model of the morse code exhibits a good deal more probabilistic structure than a first-order model, as can be seen by comparing the transition probabilities shown in Table VIII. In the interest of simplicity, however, it was decided to use a first-order model. The assumption of a first order model then means that

$$P(x_k|x_o,\ldots,x_{k-1}) = P(x_k|x_{k-1})$$

where $x_k$ is the kth character of a transmitted sequence.

The transition matrix lists the following transition probabilities:

$$P(x_k|x_{k-1}) \qquad \text{for each} \quad x_k = a_i$$

where

$a_1$ = . (dot) (VI-1a)

$a_2$ = _ (dash) (VI-1b)

$a_3$ = ^ (element-space) (VI-1c)

$a_4$ = ~ (letter-space) (VI-1d)

TABLE VIII

MARKOV TRANSITION PROBABILITIES

FIRST-ORDER MODEL

| | . | ^ | ~ | — |
|---|---|---|---|---|
| . | 0 | .682 | .318 | 0 |
| ^ | .554 | 0 | 0 | .446 |
| ~ | .5 | 0 | 0 | .5 |
| — | 0 | .684 | .316 | 0 |

THIRD-ORDER MODEL

| | . | ^ | ~ | — |
|---|---|---|---|---|
| .^. | 0 | .438 | .562 | 0 |
| .^— | 0 | .615 | .385 | 0 |
| .~. | 0 | .923 | .077 | 0 |
| .~— | 0 | .923 | .077 | 0 |
| ^.^ | .556 | 0 | 0 | .444 |
| ^.~ | .50 | 0 | 0 | .50 |
| ^—^ | .538 | 0 | 0 | .462 |
| ^—~ | .50 | 0 | 0 | .50 |
| ~.^ | .50 | 0 | 0 | .50 |
| ~.~ | .50 | 0 | 0 | .50 |
| ~—^ | .583 | 0 | 0 | .417 |
| ~—~ | .50 | 0 | 0 | .50 |
| —^. | 0 | .571 | .429 | 0 |
| —^— | 0 | .545 | .455 | 0 |
| —~. | 0 | .923 | .077 | 0 |
| —~— | 0 | .923 | .077 | 0 |

(Word spaces were not considered as separate characters, but as combinations of letter-spaces and element-spaces.)

C. SEQUENCE PROBABILITIES

Based on the first order model assumption, the probability of any particular transmitted sequence of length n is given by

$$P(x_o,x_1,\ldots,x_{n-1}) = \prod_{i=0}^{n-1} P(x_i|x_{i-1}) \qquad \text{(VI-2)}$$

Then, given an input sequence to the decoder $(z_o,\ldots,z_{n-1})$, use of Bayes' rule expresses the desired conditional probability as

$$P(x_o,\ldots,x_{n-1}|z_o,\ldots,z_{n-1})$$

$$= \frac{P(z_o,\ldots,z_{n-1}|x_o,\ldots,x_{n-1})\cdot P(x_o,\ldots,x_{n-1})}{P(z_o,\ldots,z_{n-1})} \qquad \text{(VI-3)}$$

which is the probability to be maximized.

Assuming that the thresholded output is memoryless, the conditional output sequence probability becomes:

$$P(z_o,\ldots,z_{n-1}|x_o,\ldots x_{n-1}) = \prod_{i=0}^{n-1} P(z_i|x_i) \ . \qquad \text{(VI-4)}$$

Although the thresholded output is by no means memoryless, due to the decision directed nature of the Q estimation algorithm in the preceding filter, this assumption is

nevertheless made in order to render the computation of the necessary probabilities tractable. The dependence of the thresholded character output decision on previous and future decisions will be removed to a certain extent in Section D, justifying the assumption of memorylessness at this point.

Then, using (VI-2) and (VI-4), and realizing that $P(z_o,\ldots,z_{n-1})$ is constant, maximization of (VI-3) is equivalent to maximizing the expression:

$$\prod_{i=0}^{n-1} P(x_i|x_{i-1}) \prod_{i=0}^{n-1} P(z_i|x_i) \quad .$$

Maximization of this sequence is equivalent to minimizing the negative logarithm, since ln $P(\cdot)$ is a monotonic function of $P(\cdot)$. Thus

$$L(x_o,\ldots,x_{n-1}) \triangleq - \sum_{i=0}^{n-1} [\ln P(x_i|x_{i-1}) + \ln P(z_i|x_i)]$$

is the function to be minimized by the Viterbi algorithm. An outline of how the Viterbi algorithm performs this minimization is presented in Appendix B.

## D. LIKELIHOOD COMPUTATION

The likelihoods $P(z_i|x_i)$ may be computed from the smoothed signal amplitude and received character duration as follows.

Define the following figures of merit for amplitude and duration:

$a_m$ = amplitude figure of merit for a dot-length duration

$t_m$ = time figure of merit for a dot-length duration

$A_m$ = amplitude figure of merit for a dash-length duration

$T_m$ = time figure of merit for a dash-length duration

If these figures of merit are scaled such that their values are between 0 and 1, then they may be interpreted as follows:

$a_m$ = probability that a mark occurred during a dot-length interval.

$t_m$ = probability that a mark is a dot, or

= probability that a space is an element-space.

$A_m$ = probability that a mark occurred during a dash-length interval.

$T_m$ = probability that a mark is a dash, or

= probability that a space is a letter-space.

Likelihoods, then, may be computed by utilizing these values as probabilities. For example, the probability that the thresholded output is a dot, given that the input is an element-space is simply:

$$P(z=a_1 \mid x=a_3)$$

$$= Pr(z=\text{dot} \mid z=\text{mark}) \cdot Pr(z \neq \text{dash} \mid z=\text{mark}) \cdot Pr(z=\text{mark})$$

$$= t_m \cdot (1-T_m) \cdot a_m$$

The likelihoods for each case were determined as above and are given as follows:

$$P(z=a_1|x=a_i) = a_m t_m (1 - T_m)$$

$$P(z=a_2|x=a_i) = a_m (1 - t_m (1 - T_m))$$

$$P(z=a_3|x=a_i) = (1 - a_m) t_m (1 - T_m)$$

$$P(z=a_4|x=a_i) = (1 - a_m)(1 - t_m (1 - T_m))$$

for i = 1,3 where the $a_i$ are given by (VI-1); and

$$P(z=a_1|x=a_i) = a_m (1 - T_m (1 - t_m))$$

$$P(z=a_2|x=a_i) = A_m T_m (1 - t_m)$$

$$P(z=a_3|x=a_i) = (1 - a_m)(1 - T_m (1 - t_m))$$

$$P(z=a_4|x=a_i) = (1 - A_m) T_m (1 - t_m)$$

for i = 2,4.

The figures of merit $a_m$, $A_m$, $t_m$, $T_m$ were obtained by using the thresholded, smoothed output to make "tentative" decisions, and then computing the merit of these decisions. The tentative decisions were:

1) If $x_{out} = 1$ and measured duration $\leq 2T$, then $x_k$ = dot

2) If $x_{out} = 1$ and measured duration $> 2T$, then $x_k$ = dash

3) If $x_{out} = 0$ and measured duration $\leq 2T$, then $x_k$ = element-space

4) If $x_{out} = 0$ and measured duration $> 2T$, then $x_k$ = letter-space

For $x_k$ = dot or element-space, the previous T values of smoothed output were averaged and the amplitude figure of merit was taken as

$$a_m = \begin{cases} 0 & -\frac{\hat{a}}{2} \geq \bar{x}_s(T) \\ \bar{x}_s(T)/\hat{a} + 0.5 & \frac{\hat{a}}{2} \geq \bar{x}_s(T) \geq -\frac{\hat{a}}{2} \\ 1 & \bar{x}_s(T) \geq \frac{\hat{a}}{2} \end{cases}$$

where

$\bar{x}_s(T)$ = the average of the smoothed output, $x_s$, over the previous T seconds.

$\hat{a}$ = amplitude of smoothed output signal.

The time figures of merit were

$$t_m = \begin{cases} 1 & 0 \leq t_d \leq T \\ 2 - (t_d + T)/2T & T \leq t_d \leq 3T \\ 0 & 3T \leq t_d \end{cases}$$

$$T_m = \begin{cases} 0 & 0 \leq t_d \leq T_d/3 \\ 1 - (T_d - t_d)/(0.67T_d) & T_d/3 \leq t_d \leq T_d \\ 1 & T_d \leq t_d \end{cases}$$

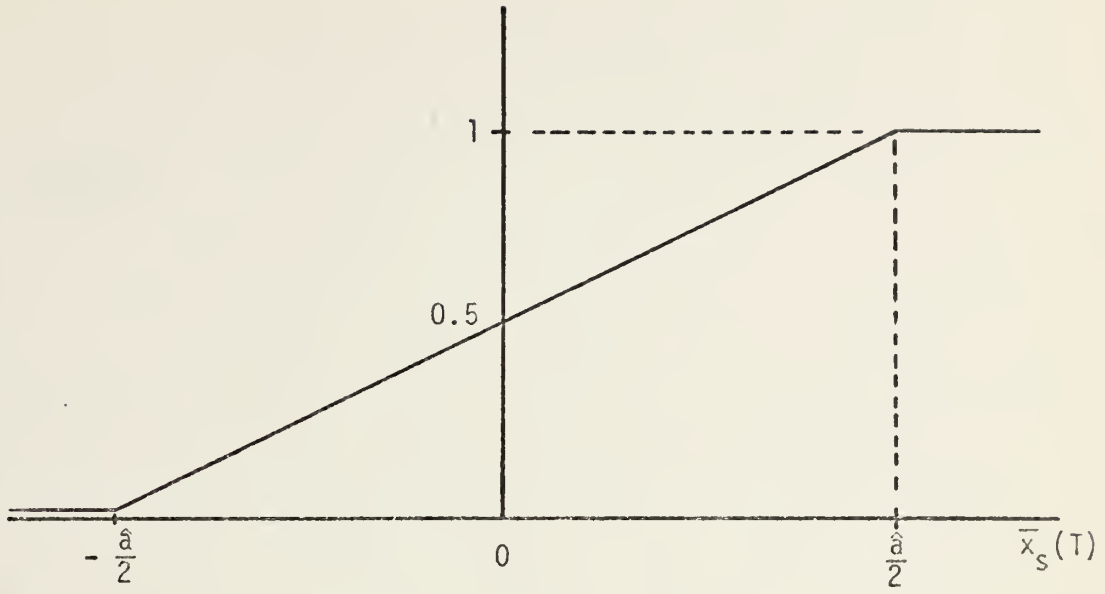
where $t_d$ = measured duration of character duration.

These functions are sketched in Figure 22. For $x_k$ = dash or letter-space, the previous $T_d$ values of $x_s$ were averaged to obtain the value for $A_m$:

$$A_m = \begin{cases} 0 & -\frac{\hat{a}}{2} \geq \bar{x}_s(T_d) \\ \bar{x}_s(T_d)/\hat{a} + 0.5 & \frac{\hat{a}}{2} \geq \bar{x}_s(T_d) \geq -\frac{\hat{a}}{2} \\ 1 & \bar{x}_s(T_d) \geq \frac{\hat{a}}{2} \end{cases}$$

These likelihood computations allow decisions to be made by the Viterbi algorithm to determine the most likely character on a character-versus-character basis. It was necessary, however, to extend these computations to cover more complicated situations such as that depicted in Figure 23. The thresholded output shown decodes as .^.^. , although the sequence .^— was actually transmitted. The Viterbi algorithm implemented using the above likelihood computations decoded the sequence as .^—^. , however, since no provision was made to account for the non-memoryless nature of the thresholded output.

a) Amplitude Figure of Merit




b) Duration Figures of Merit

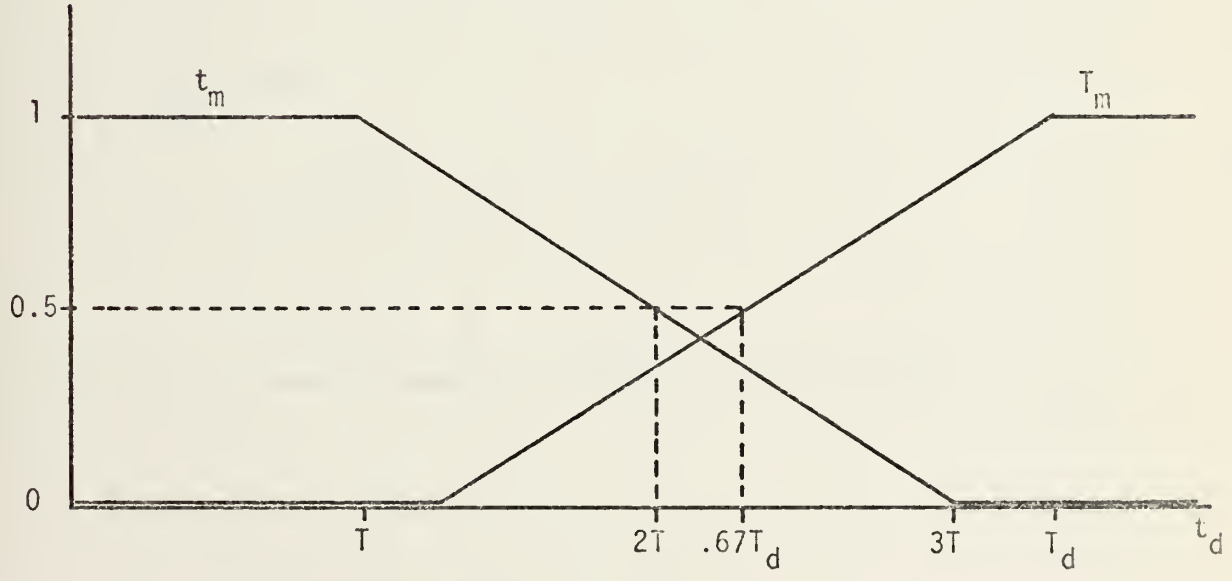



Figure 22. Sketch of Figures of Merit

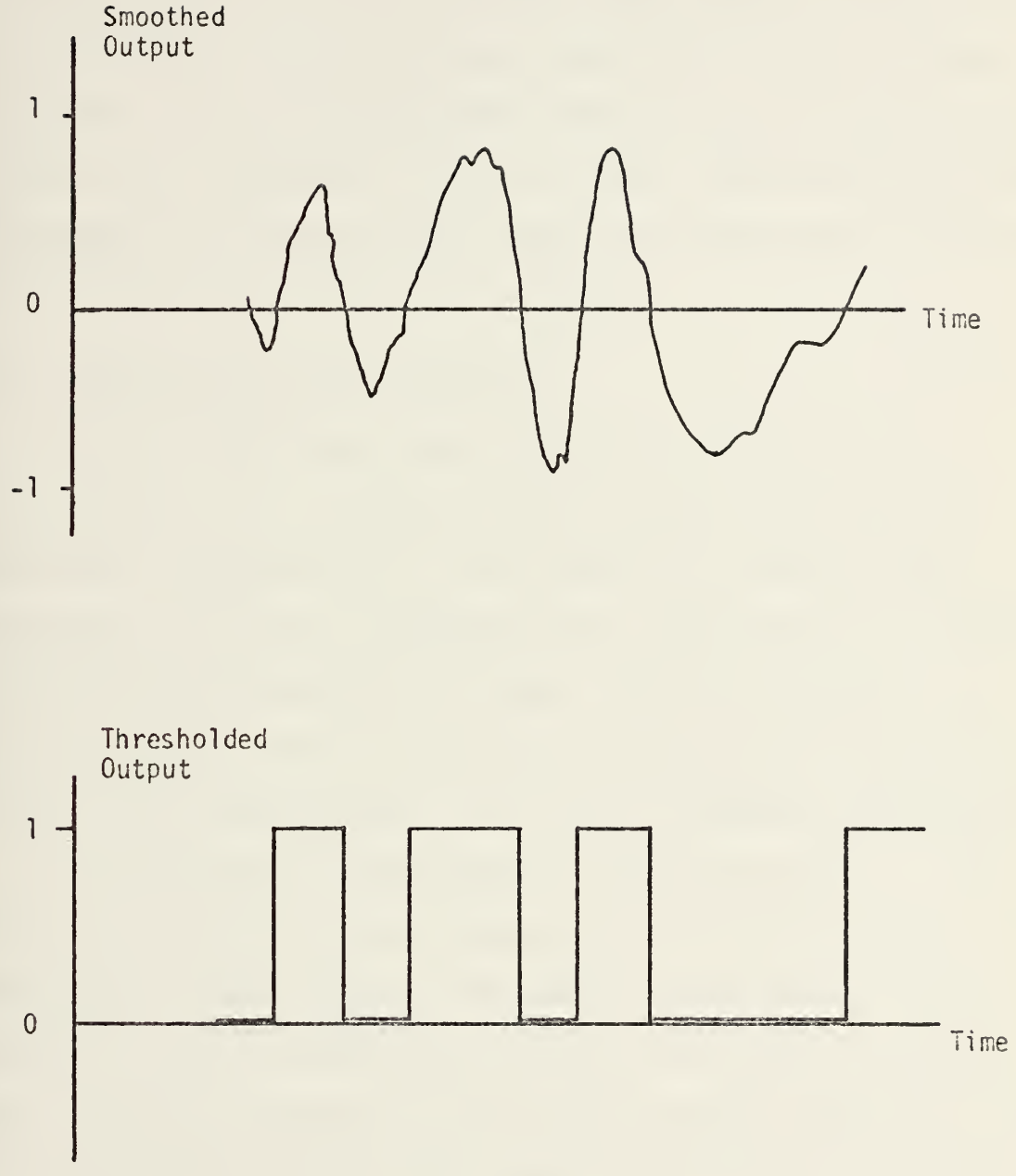



Figure 23. Illustration of Signal Requiring Modified Likelihood Computation

Two modifications were made, therefore, in order to account for situations such as this: First, when a dash or letter-space was decoded by the thresholded output, this character was divided into three equal segments and dot and element-space likelihoods were computed for each segment. Secondly, when the thresholded output consecutively decoded three short characters, then the total duration of all three characters was obtained and dash and letter-space likelihoods were computed for the total interval.

## E. IMPLEMENTATION AND RESULTS

The algorithm was coded in Fortran and run as a subroutine of the original Kalman filter and smoothing programs. Sequences of 9-11 elements in duration were decoded. In order to determine the contribution of context information supplied by the first order transition probabilities, the algorithm was also run with equiprobable transition probabilities, i.e. as a ML sequence estimator.

The results are summarized in Tables IX and X. Again both bit and letter error rates are presented with the hand-decoded smoothed output repeated here for ease of comparison. Table XI shows a comparison of bit error rates for ML and MAP estimation showing a slight improvement provided by the MAP estimation, taking advantage of transition probabilities.

The reduction in error rate, although not as great as hoped for, shows that the thresholded output bit error rate can be improved significantly at the higher SNR's, but

TABLE IX

VITERBI DECODER ERROR RATES - 35 wpm

$\overline{AR}$ SEQUENCE, PERFECT CODE:

| SNR (2 kHz) NO PRE-FILTER (db) | SMOOTHED OUTPUT BIT (%) | SMOOTHED OUTPUT LTR (%) | VITERBI OUTPUT BIT (%) | VITERBI OUTPUT LTR (%) | IMPROVEMENT RATIO IN BIT ERROR RATE |
|---|---|---|---|---|---|
| 6 | 0.83 | 9 | .35 | 3 | 2.4 |
| 5 | 3.0 | 27 | 2.5 | 23 | 1.2 |
| 4 | 6.7 | 49 | 6.8 | 49 | - |
| 3 | 35 | 96 | 35 | 100 | - |

TYPICAL SEQUENCE, PERFECT CODE:

| SNR (2 kHz) NO PRE-FILTER (db) | SMOOTHED OUTPUT BIT (%) | SMOOTHED OUTPUT LTR (%) | VITERBI OUTPUT BIT (%) | VITERBI OUTPUT LTR (%) | IMPROVEMENT RATIO IN BIT ERROR RATE |
|---|---|---|---|---|---|
| 6 | 0.70 | 8.0 | 0.33 | 3 | 2.1 |
| 5 | 1.2 | 15 | 0.90 | 11 | 1.3 |
| 4 | 4.5 | 34 | 4.4 | 34 | - |
| 3 | 30 | 92 | 30 | 96 | - |

TABLE X

VITERBI DECODER ERROR RATES - 25 wpm

$\overline{AR}$ SEQUENCE, PERFECT CODE:

| SNR (2kHz) NO PRE-FILTER (db) | SMOOTHED OUTPUT BIT (%) | SMOOTHED OUTPUT LTR (%) | VITERBI OUTPUT BIT (%) | VITERBI OUTPUT LTR (%) | IMPROVEMENT RATIO IN BIT ERROR RATE |
|---|---|---|---|---|---|
| 6 | .77 | 10 | 0.31 | 2 | 2.5 |
| 5 | 1.3 | 17 | 1.0 | 13 | 1.3 |
| 4 | 1.7 | 18 | 1.5 | 17 | 1.1 |
| 3 | 4.5 | 36 | 4.4 | 38 | - |

TYPICAL SEQUENCE, PERFECT CODE:

| SNR (2kHz) NO PRE-FILTER (db) | SMOOTHED OUTPUT BIT (%) | SMOOTHED OUTPUT LTR (%) | VITERBI OUTPUT BIT (%) | VITERBI OUTPUT LTR (%) | IMPROVEMENT RATIO IN BIT ERROR RATE |
|---|---|---|---|---|---|
| 6 | 0.70 | 8 | 0.37 | 3 | 1.9 |
| 5 | 1.1 | 12 | 0.80 | 10 | 1.4 |
| 4 | 1.2 | 15 | 0.90 | 13 | 1.3 |
| 3 | 2.9 | 24 | 2.7 | 22 | 1.1 |

TABLE XI

COMPARISON OF ML AND MAP ESTIMATES

| SNR (2kHz) NO PRE-FILTER (db) | MAXIMUM LIKELIHOOD ESTIMATOR BIT ERROR RATE (%) | MAXIMUM A-POSTERIORI ESTIMATOR BIT ERROR RATE (%) | IMPROVEMENT RATIO |
|---|---|---|---|
| 6 | 0.45 | 0.37 | 1.2 |
| 5 | 0.91 | 0.80 | 1.1 |
| 4 | 1.1 | 0.90 | 1.2 |
| 3 | 2.8 | 2.7 | 1.0 |

Typical sequence 25 wpm, perfect code

bit error rates of about 3% or higher remain practically unaffected. Again, it was noted that most errors are isolated dot insertions during word-spaces. Since the third order model accounts for the small transition probability of such events, such a model should virtually eliminate this type of error, resulting in a very significant decrease in error rates.

Table XII shows a typical decoded sequence as output by the algorithm, with the filtered and smoothed outputs shown for comparison. This sequence was part of the 35 wpm $\overline{AR}$ sequence at 5 dB SNR in 2 kHz. Since it was noted that the algorithm created some new errors as well as correcting errors, likelihood computations could probably be improved by employing better character-length density estimation procedures.

Two examples of likelihood computations and the evolution of the most likely sequence as the algorithm progresses are shown in Tables XIIIa and b. Table XIIIa is an example of decoding with highly probable likelihoods, while Table XIIIb shows an example of a correction by the algorithm. The array of numbers headed by SURVIVOR SEQUENCES indicate to which previous node the node at stage k is connected; the length of each survivor sequence is shown immediately below this array. The line through the survivor sequence array shows the final path for the minimum-length sequence. The next array shows the computed log-likelihoods for each character,

followed by the minimum length sequence with its length. (The number 200 was sufficiently large to be used for infinity.) Finally, the mark and space durations as determined by the thresholded output are shown for comparison.

In these tables (XIIIa and b), the following numbers correspond to the decoded morse characters:

1 - dot

2 - element-space

3,4 - letter-space

5,6 - dash

The letter-space and dash are represnted by two numbers since such decisions may result either from comparisons made on a straightforward character versus character basis or from comparisons made utilizing the modified likelihood computation previously described. In each case the sequence ^.^—^.~ was transmitted; Table XIIIb shows how the Viterbi algorithm corrected the thresholded sequence (^.^.^.^.~) to form a dash from the inner .^. sequence. This situation is similar to that shown previously in Figure 23.

TABLE XII

COMPARISON OF VITERBI OUTPUT WITH
SMOOTHED AND FILTERED OUTPUTS

| | |
|---|---|
| FILTERED | $\overline{AR}$ E $\overline{AR}$ $\overline{AR}$ $\overline{UC}$ $\overline{AL}$ $\overline{AR}$ $\overline{AR}$ E $\overline{AR}$ E $\overline{UR}$ $\overline{AR}$ $\overline{AR}$ |
| SMOOTHED | $\overline{AR}$ E $\overline{MR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ E $\overline{AR}$ E $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ |
| VITERBI | $\overline{AR}$ E $\overline{AR}$ $\overline{AR}$ $\overline{MR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ E $\overline{AN}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ |
| | |
| FILTERED | $\overline{RR}$ $\overline{AR}$ E $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ I $\overline{AG}$ |
| SMOOTHED | $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ I $\overline{AG}$ |
| VITERBI | $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ $\overline{AR}$ |

Note :

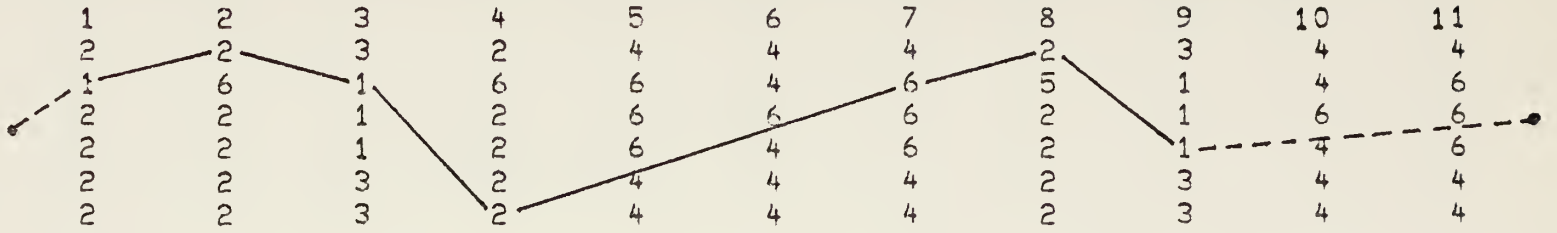
1. Sequence of $\overline{AR}$ at 35 wpm, 5 db SNR, tranmitted.

SUMMARY OF VITERBI OUTPUT:

| | |
|---|---|
| New errors made | 2 |
| Errors uncorrected | 2 |
| Corrections | 4 |
| Net improvement | 2 |

TABLE XIIIa. VITERBI DECODER COMPUTATION

SURVIVOR SEQUENCES



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 2 | 4 | 4 | 4 | 2 | 3 | 4 | 4 |
| 2 | 1 | 6 | 1 | 6 | 6 | 4 | 6 | 5 | 1 | 4 | 6 |
| 3 | 2 | 2 | 1 | 2 | 6 | 6 | 6 | 2 | 1 | 6 | 6 |
| 4 | 2 | 2 | 1 | 2 | 6 | 4 | 6 | 2 | 1 | 4 | 6 |
| 5 | 2 | 2 | 3 | 2 | 4 | 4 | 4 | 2 | 3 | 4 | 4 |
| 6 | 2 | 2 | 3 | 2 | 4 | 4 | 4 | 2 | 3 | 4 | 4 |

SURVIVOR LENGTHS

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 201.08 | 2.56 | 210.79 | 203.63 | 405.00 | 405.09 | 207.26 | 6.74 | 212.75 | 407.54 | 409.07 |
| 2 | 1.31 | 201.24 | 2.94 | 401.46 | 402.62 | 405.09 | 5.38 | 208.47 | 207.13 | 407.54 | 406.68 |
| 3 | 5.68 | 9.41 | 5.28 | 205.25 | 403.39 | 411.95 | 10.74 | 11.37 | 207.89 | 408.42 | 407.46 |
| 4 | 201.08 | 203.62 | 203.71 | 5.88 | 403.39 | 405.09 | 206.15 | 207.68 | 7.89 | 407.54 | 407.46 |
| 5 | 201.08 | 7.73 | 210.79 | 203.87 | 405.00 | 405.09 | 207.26 | 9.85 | 212.75 | 407.54 | 409.07 |
| 6 | 201.08 | 202.24 | 210.79 | 5.00 | 405.00 | 405.09 | 207.26 | 206.30 | 212.75 | 407.54 | 409.07 |

LIKELIHOODS

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 200.00 | .56 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 | .68 | 200.00 | 200.00 | 200.00 |
| 2 | -.00 | .86 | -.00 | 200.00 | 200.00 | 200.00 | -.00 | .82 | 200.00 | 200.00 | 200.00 |
| 3 | 4.60 | 5.79 | 1.57 | 200.00 | 200.00 | 200.00 | 4.59 | 3.69 | 200.00 | 200.00 | 200.00 |
| 4 | 200.00 | 200.00 | 200.00 | .63 | 200.00 | 200.00 | 200.00 | 200.00 | -.00 | 200.00 | 200.00 |
| 5 | 200.00 | 5.49 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 | 3.55 | 200.00 | 200.00 | 200.00 |
| 6 | 200.00 | 200.00 | 200.00 | 1.13 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 |

| MARK | SPACE |
|---|---|
| 0 | 40 |
| 40 | 0 |
| 0 | 46 |
| 114 | 0 |
| 0 | 40 |
| 42 | 0 |
| 0 | 120 |

| LENGTH | DECODED SEQUENCE |
|---|---|
| 7.889 | 2126214 |

TABLE XIIIb. VITERBI DECODER COMPUTATION

SURVIVOR SEQUENCES

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 2 | 3 | 2 | 4 | 2 | 2 | 4 | 4 |
| 2 | 1 | 6 | 1 | 6 | 1 | 1 | 6 | 4 | 1 | 6 | 6 |
| 3 | 2 | 2 | 1 | 2 | 1 | 2 | 6 | 2 | 1 | 6 | 6 |
| 4 | 2 | 2 | 1 | 2 | 1 | 2 | 6 | 2 | 1 | 4 | 6 |
| 5 | 2 | 2 | 3 | 2 | 3 | 2 | 4 | 2 | 2 | 4 | 4 |
| 6 | 2 | 2 | 3 | 2 | 3 | 2 | 4 | 2 | 2 | 4 | 4 |

SURVIVOR LENGTHS

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 201.08 | 2.57 | 208.42 | 4.48 | 10.73 | 6.88 | 207.84 | 7.96 | 408.45 | 408.79 | 410.32 |
| 2 | 1.31 | 201.42 | 2.96 | 202.40 | 5.08 | 12.02 | 6.63 | 207.76 | 208.34 | 408.22 | 407.94 |
| 3 | 3.99 | 7.03 | 203.72 | 7.72 | 205.63 | 9.30 | 207.41 | 208.94 | 209.11 | 408.99 | 408.71 |
| 4 | 201.08 | 203.62 | 203.72 | 6.45 | 205.63 | 207.39 | 207.41 | 208.94 | 9.11 | 408.79 | 408.71 |
| 5 | 201.08 | 5.19 | 208.42 | 6.23 | 209.10 | 8.13 | 207.84 | 207.56 | 408.69 | 408.79 | 410.32 |
| 6 | 201.08 | 202.24 | 208.42 | 6.25 | 209.10 | 206.01 | 207.84 | 207.56 | 408.69 | 408.79 | 410.32 |

LIKELIHOODS

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 200.00 | .58 | 200.00 | .84 | 1.62 | 1.11 | 200.00 | .64 | 200.00 | 200.00 | 200.00 |
| 2 | -.00 | 1.04 | -.00 | .95 | .22 | .91 | -.00 | .75 | 200.00 | 200.00 | 200.00 |
| 3 | 2.91 | 3.42 | 200.00 | 2.46 | 200.00 | 1.92 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 |
| 4 | 200.00 | 200.00 | 200.00 | 1.19 | 200.00 | 200.00 | 200.00 | 200.00 | -.00 | 200.00 | 200.00 |
| 5 | 200.00 | 2.95 | 200.00 | 2.35 | 200.00 | 2.12 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 |
| 6 | 200.00 | 200.00 | 200.00 | 2.37 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 | 200.00 |

| MARK | SPACE |
|---|---|
| 0 | 42 |
| 44 | 0 |
| 0 | 38 |
| 48 | 0 |
| 0 | 20 |
| 52 | 0 |
| 0 | 40 |
| 40 | 0 |
| 0 | 120 |

| LENGTH | DECODED SEQUENCE |
|---|---|
| 9.107 | 2126214 |

## VII. PRE-DETECTION MODEL AND FILTER

Since it is assumed that the receiver has been tuned to the signal carrier frequency, it is sufficient to model the signal as one of known frequency and unknown amplitude and phase of the form $A \sin(\omega t+\theta)$. The received signal then is of the form

$$z(t) = A \sin(\omega t+\theta) + v(t)$$

where $v(t)$ represents additive white gaussian noise. A state model of the signal process may be obtained by rewriting the transmitted signal in the form $a \sin \omega t + b \cos \omega t$ [4]. Letting $x_1$ represent the discretized signal in this form, the following state model results:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \omega\tau \\ -\omega\tau & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

where $x_2(t) \triangleq b \cos \omega t - a \sin \omega t$

$k$ = time index

$\tau$ = sampling interval

The observation model then is

$$z(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + v(k) \quad .$$

This model may be written more compactly in vector notation:

$$\underline{x}(k+1) = \underline{\Phi}(k)\ \underline{x}(k)$$

$$z(k) = \underline{H}(k)\ \underline{x}(k) + v(k)$$

where $\underline{\Phi}$ is the state transition matrix and $\underline{H}$ is the measurement vector. The on-off keying nature of the signal may be accounted for in an intuitive way simply by multiplying the observed value, $z(k)$, by the probability that the signal is present. This probability is readily obtained from the demodulated output of the filter by using the algorithm previously derived for mark and space transition probabilities for the baseband signal.

## B. FILTER ALGORITHM

The general filter algorithm presented in Appendix A, using the above signal model, was used to filter the (down-converted) IF signal. Demodulation was accomplished digitally by squaring and averaging the $x_1$ state estimate. The subroutine used previously for the calculation of Q was then used in exactly the same way as for the baseband model except

that the indices k and j were advanced by an amount of time equal to the delay due to averaging in the demodulation process. Additionally it was found that using zero probability during space intervals was too low to allow recovery when the signal pulse occurred. The probability 0.5 was found to be sufficient and was used whenever zero probability would normally have been called for.

C. IMPLEMENTATION AND RESULTS

This filter was programmed and tested using test signals of -2 dB though +2 dB SNR in a 2 kHz bandwidth, and -15 dB through -11 dB in a 2 kHz bandwidth with 100 Hz bandpass filtering prior to sampling. In order to determine the effectiveness of modifying z(k) by the probability as described above, the filter was also run with the probability set to 1. The signal was sampled at 4000 samples per second, the (down-converted) carrier frequency was 1000 Hz, and the modulating signal was a square wave with period equivalent to a code speed of 25 wpm. Because of the lengthy processing time (1 second of data required about 30 seconds of processing time), no large sample error rates as such were obtained. However, as can be seen from the output, Figures 24 through 31, the filter performed well on signals of SNR -1 dB and above (2 kHz) and -14 dB and above with 100 Hz pre-filtering.

Figures 24 through 27 are typical examples of the test run made using the straightforward filter with no modification

of the observed signal. In these figures channel 1 is the input signal, channel 2 is the filtered signal, and channel 3 is the demodulated output. Figures 24 and 25 are the results obtained from the signal in the 2 kHz bandwidth; Figures 26 and 27 show the results of 100 Hz analog bandpass filtering prior to sampling.

Figures 28 through 31 are the outputs of the processor using the modified observation model. Channel 1 is the input signal; channel 2 is the input multiplied by the probability obtained from the transition probability estimation algorithm; channel 3 is the filtered signal, and channel 4 is the demodulated output. The input signals were the same as those shown in Figures 24-27.

Table XIV presents the results of a bit error rate analysis made on a sample size of approximately 100 bits. Since the test signal was not morse code, no letter error rates were obtained. The projected letter error rates shown in the table were determined simply by multiplying the bit error rate by 10 since this is approximately the proportionality factor between these two error rates, as can be seen from Tables III-VI.

Based on this limited data, the conclusion may be tentatively drawn that a processing scheme employing 100 Hz analog filtering followed by discrete optimum linear filtering and detection will yield acceptable decoded error rates on

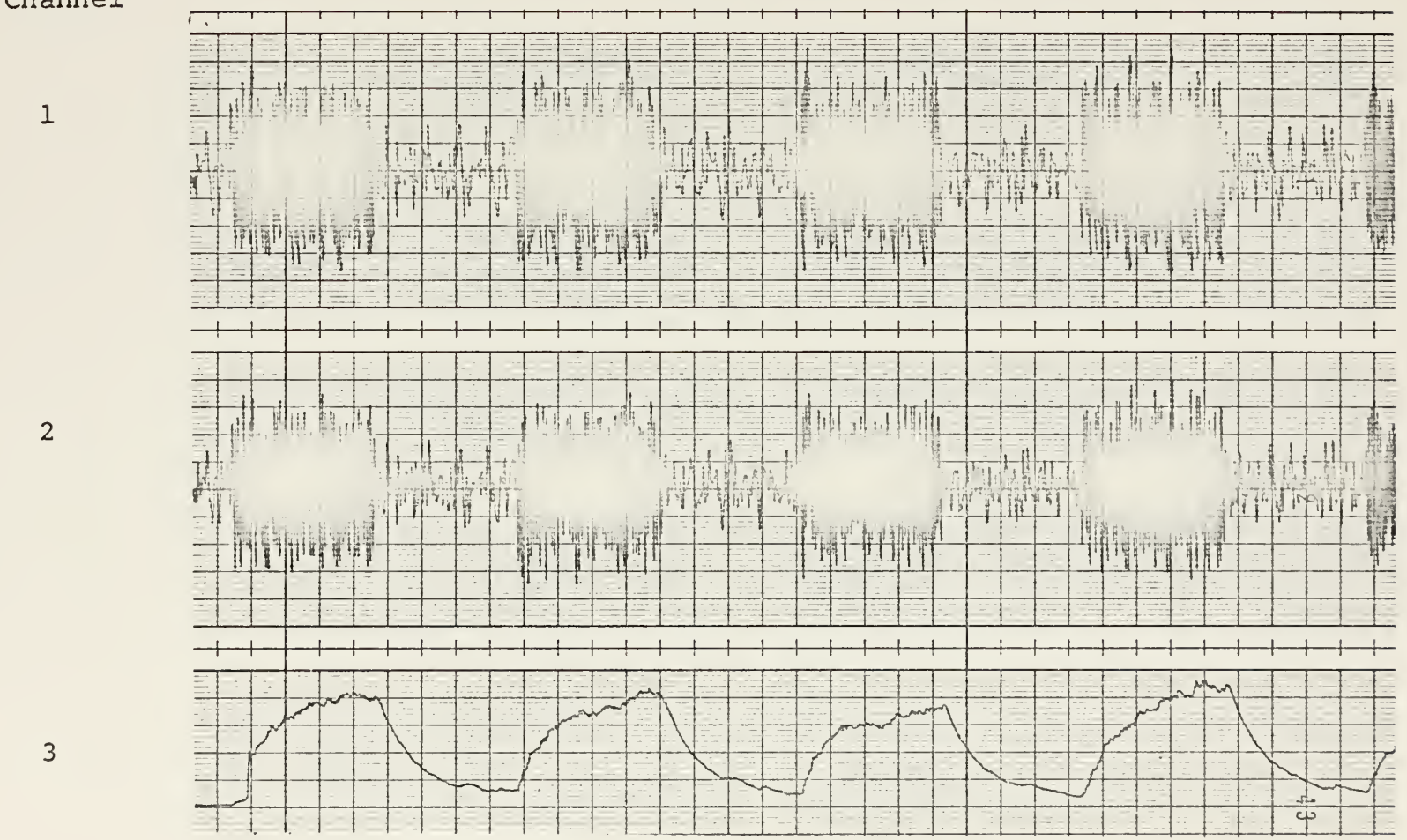



FIGURE 24. Unmodified Pre-Detection Filter Output, 2 dB SNR, 2 kHz BW

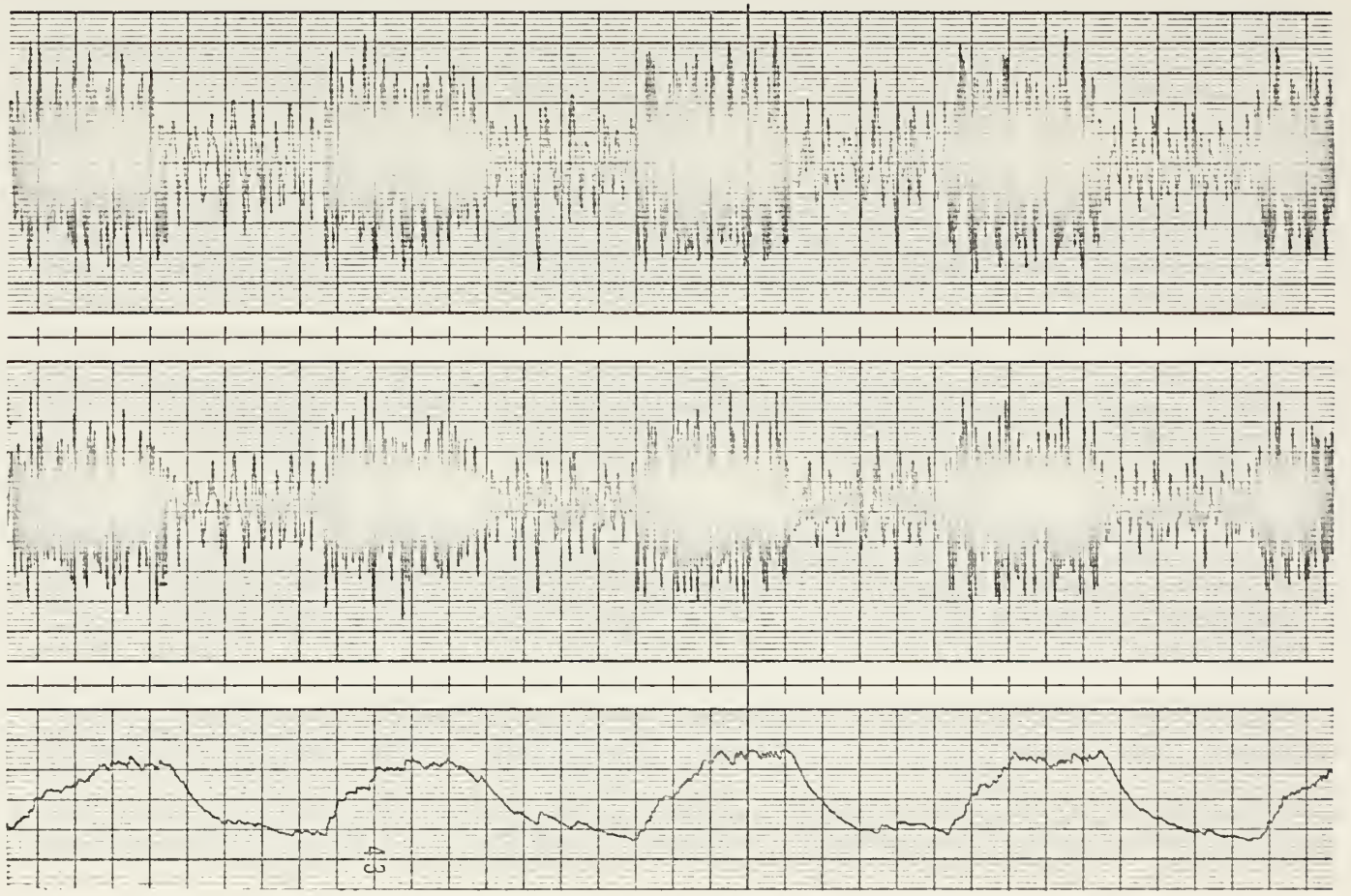



FIGURE 25. Unmodified Pre-Detection Filter Output, -1 dB SNR, 2 kHz BW

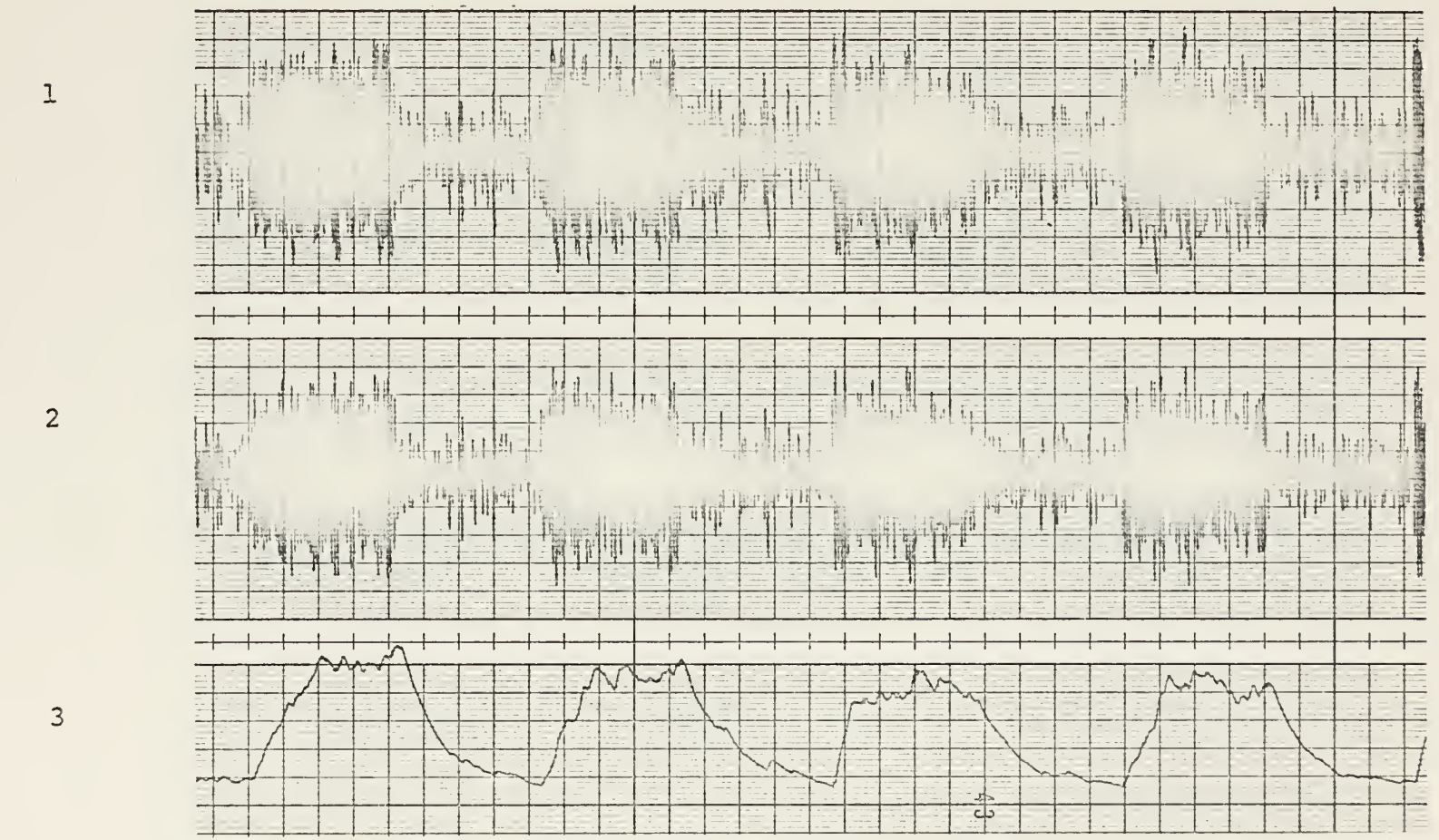



FIGURE 26. Unmodified Pre-Detection Filter Output, -11 dB SNR, 100 Hz BPF

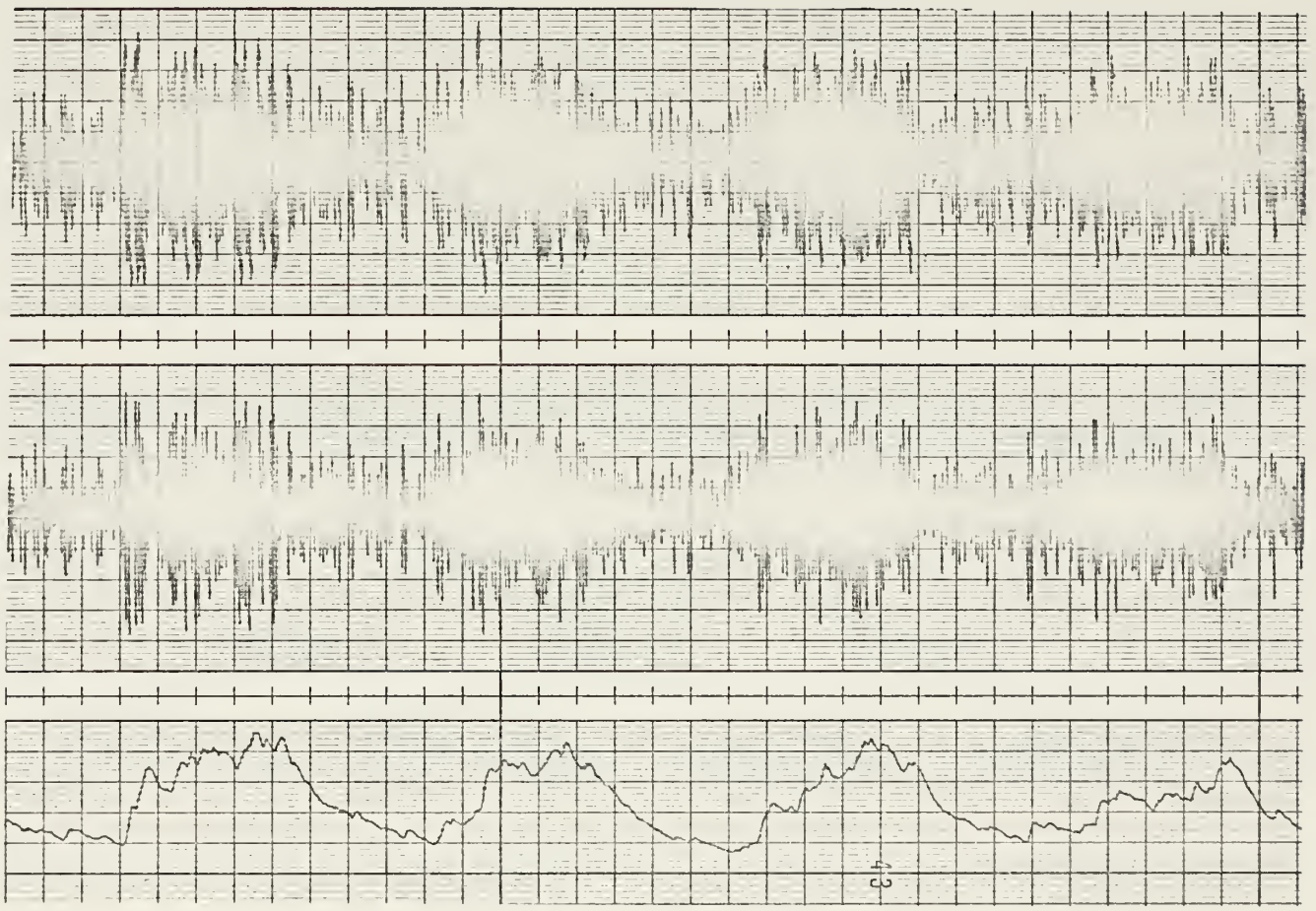



FIGURE 27. Unmodified Pre-Detection Filter Output, -14 dB SNR, 100 Hz BPF

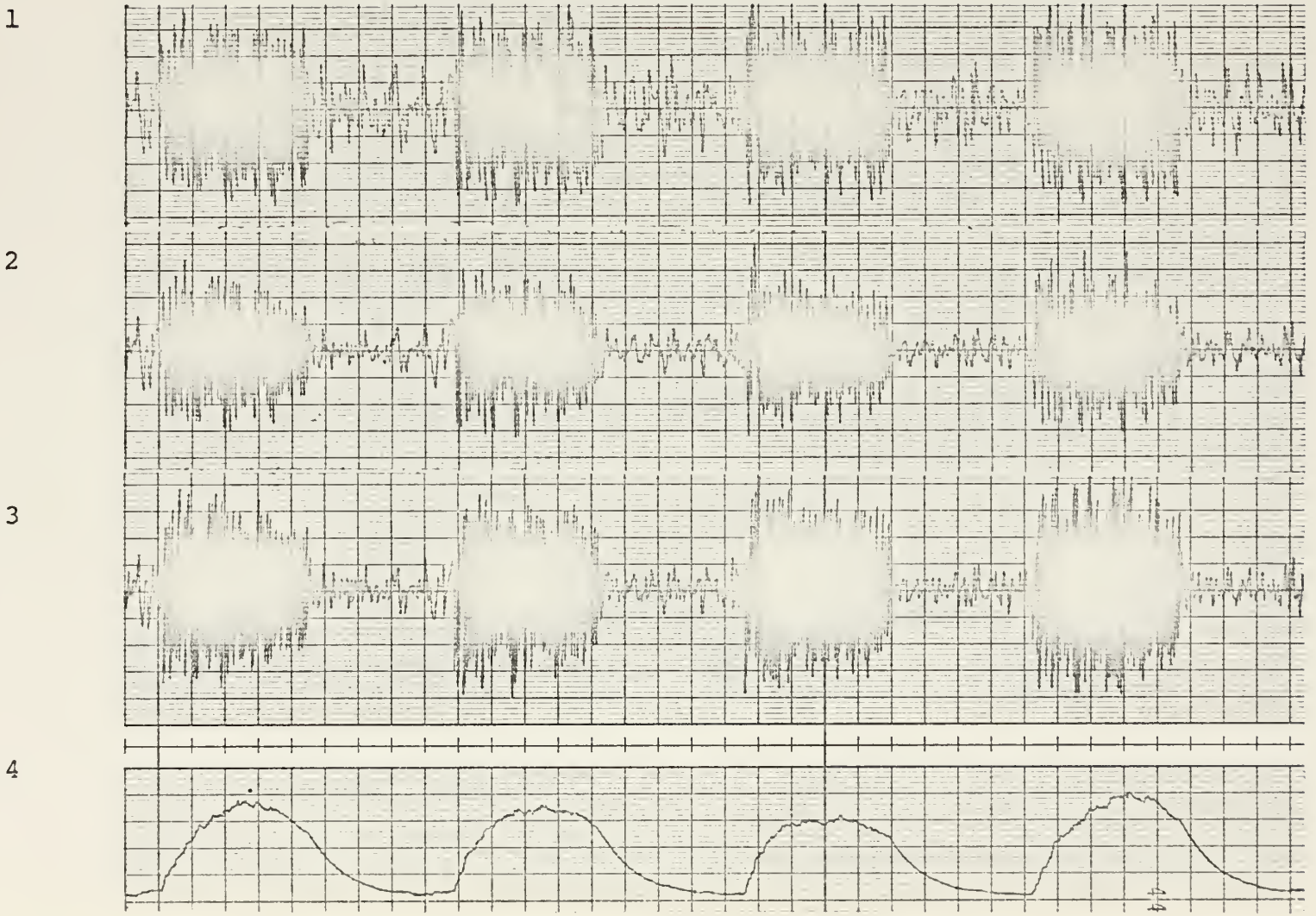



FIGURE 28. Modified Pre-Detection Filter Output, 2 dB SNR, 2 kHz BW

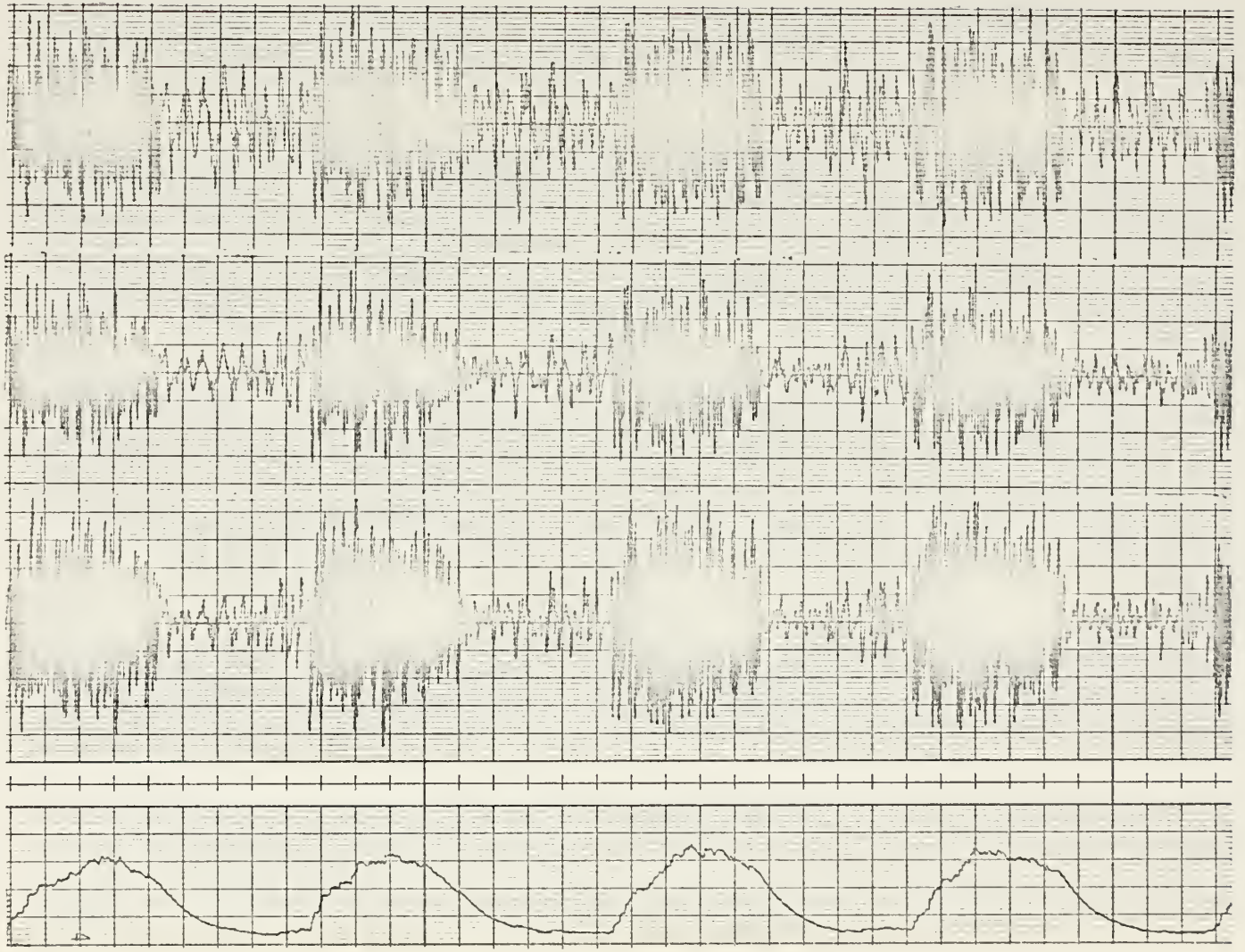



FIGURE 29. Modified Pre-Detection Filter Output, -1 dB SNR, 2 kHz BW

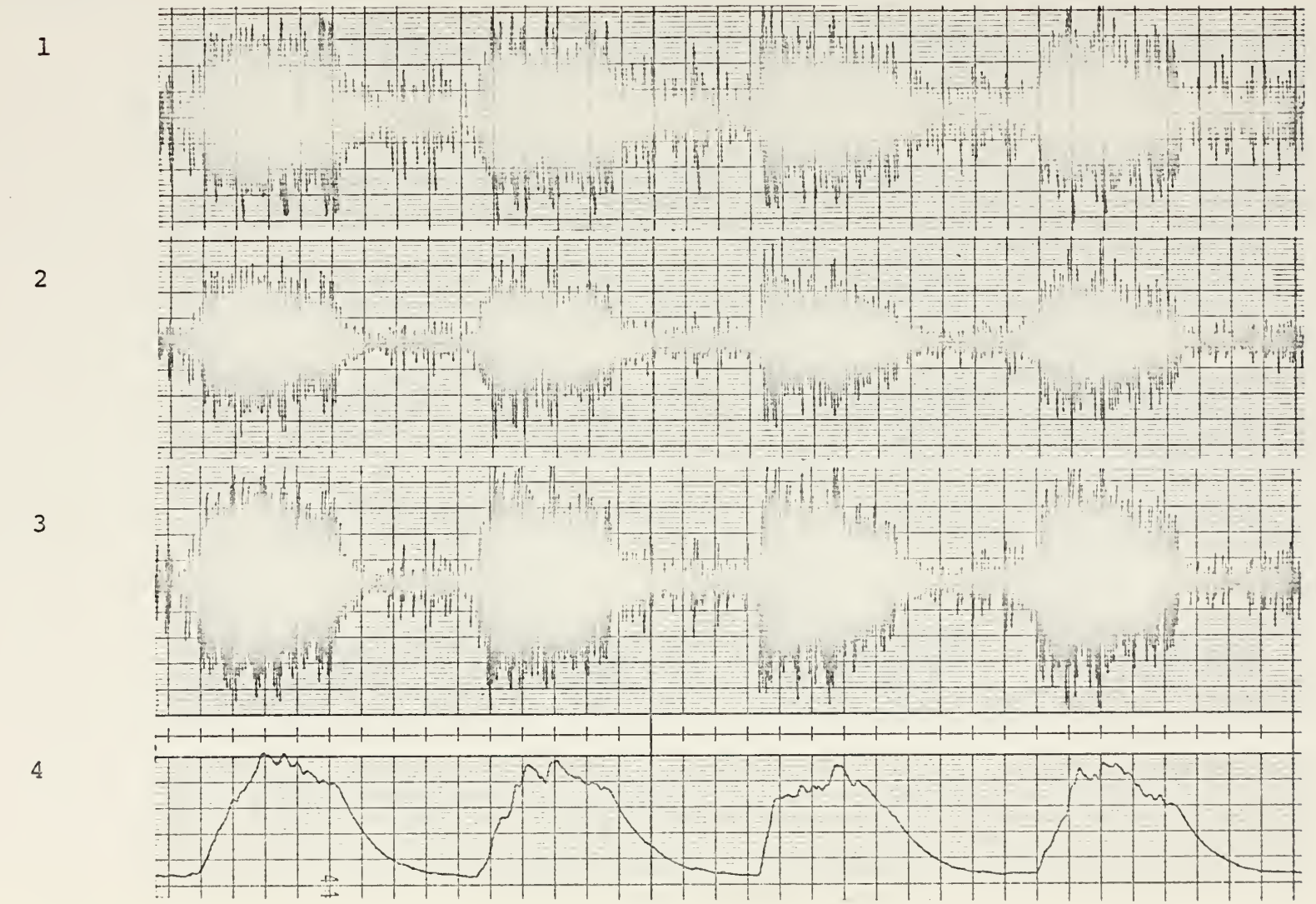



FIGURE 30. Modified Pre-Detection Filter Output, -11 dB SNR, 100 Hz BPF

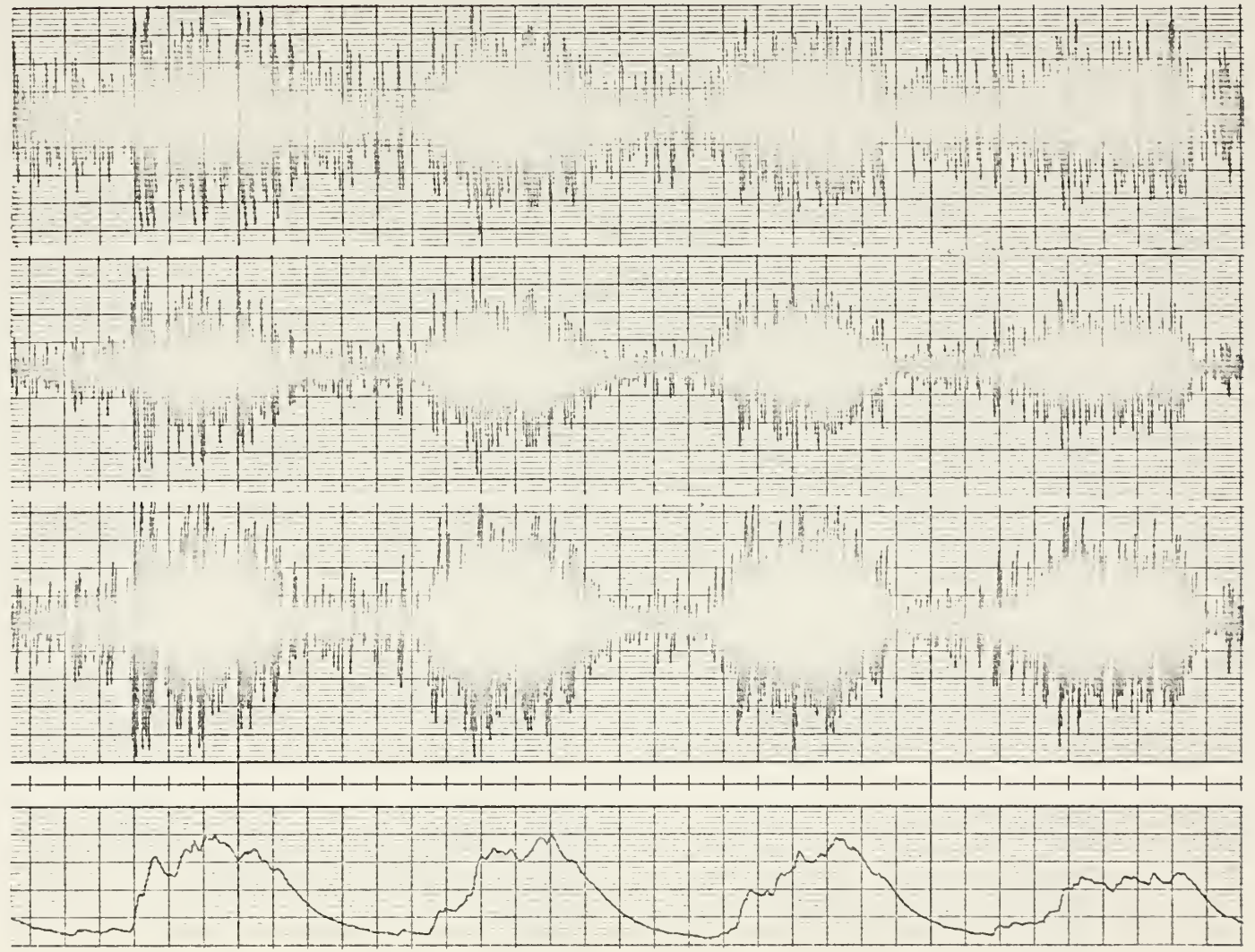



FIGURE 31. Modified Pre-Detection Filter Output, -14 dB SNR, 100 Hz BPF

TABLE XIV

ERROR RATES FOR PRE-DETECTION FILTERING

(a) No 100 Hz BP Filter:

| SNR (2 kHz) (dB) | Bit Error Rate (%) | Projected Letter Error Rate (%) |
|---|---|---|
| 2 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |
| -1 | 1 | 10 |
| -2 | 2 | 20 |

(b) With 100 Hz BP Filter:

| SNR (2 kHz) (dB) | Bit Error Rate (%) | Projected Letter Error Rate (%) |
|---|---|---|
| -11 | 0 | 0 |
| -12 | 0 | 0 |
| -13 | 0 | 0 |
| -14 | 1 | 10 |
| -15 | 1 | 10 |

NOTE: Rates are based on small sample size of approximately 100

signals of SNR as low as -14 dB in a 2 kHz bandwidth. Smoothing of the state estimates at this processing stage would probably yield additional gain although an additional penalty in processing time would be incurred.

Although this scheme meets the design objectives specified for the processor, it is probably unrealistic to assume that such processing could be implemented in real time on any existing machine without parallel processing. Coding efficiency could undoubtedly be improved but the inherent number of multiplications involved in the filter algorithm would eventually limit the processing speed. One possible alternative is to down-convert the received signal to the 0-100 Hz band, and with a 100 Hz low-pass filter in place, sample at 200 samples/sec. With such filtering, the processing described above would be feasible.

# VIII. CONCLUSIONS AND RECOMMENDATIONS

## A. CONCLUSIONS

The results of the various processing schemes considered in this investigation lead to the following definite conclusions.

1. Kalman filtering of the demodulated morse signal provides at least a 50% reduction in letter error rate over unprocessed letter error rates of approximately 60% or less.

2. Smoothing of the demodulated signal provides a reduction in letter error rate of approximately 30% over filtering alone, resulting in an overall reduction of 65% over the unprocessed error rate.

3. Viterbi decoding of the morse characters, using the smoothed state estimate for likelihood computation and using a first order Markov model of the code to obtain transition probabilities, provides a further significant reduction in error rate if the input error rate is on the order of 10% or less. For input error rates larger than 10%, Viterbi decoding is of little value.

4. In almost all cases, the processor performed better on sloppy code and typical sequences than on perfect code and $\overline{AR}$ sequences. This behavior is to be expected since the statistics of the input signal should match those for which the processor was designed in order to achieve optimum performance.

5. By using a 100 Hz bandpass filter, pre-detection Kalman filtering allows recovery of the code with a resulting error rate of about 10% or less for signals whose SNR in a 2 kHz bandwidth is -14 dB or higher.

6. The Pickering 230-D decoder (see Appendix C) allows recovery of the code with a resulting error rate of about 10% or less for signals whose SNR in a 2 kHz bandwidth is approximately -6 dB or higher, also by using a 100 Hz bandpass filter.

By making reasonable extrapolations to various other processing arrangements based on the results presented here, the projected performance of several alternative processors may be obtained with a high degree of confidence. The processing arrangements presented in Table XV are listed more or less in order of increasing complexity and power, showing the letter error rates to be expected from each; the projected figures are indicated by an asterisk. The rates presented assume that the processors, except for the human operators, are preceded by a 100 Hz bandpass filter.

Since no measurements were made for SNR's above -6 dB, except for the Pickering 230-D, the projected rates in the -4 dB column were all determined by extrapolation of the measured values. The projected values for coherent demodulation (row 6) are based on the presumption that such demodulation would provide approximately 3 dB of processing gain over non-coherent demodulation; the subsequent values for Kalman filtering, smoothing, and Viterbi decoding (rows 7-9) were

TABLE XV

PERFORMANCE OF ALTERNATIVE PROCESSING SCHEMES

| Processor | SNR (in 2 kHz) (dB) -15 | -14 | -13 | -10 | -7 | -4 |
|---|---|---|---|---|---|---|
| 1. Non-coherent demodulation | >50 | >50 | >50 | >50 | 35 | 10* |
| 2. (1) followed by Kalman Filter | >50 | >50 | >50 | >50 | 15 | 5* |
| 3. (1) followed by smoothing | >50 | >50 | >50 | >50 | 9 | 3* |
| 4. (3) followed by Viterbi Decoder | >50 | >50 | >50 | >50 | 3 | 1* |
| 5. Pickering 230-D | >50 | >50 | >50 | ~50 | 15 | 7 |
| 6. Coherent Demodulation | >50* | >50* | >50* | 35* | 10* | 5* |
| 7. (6) followed by Kalman Filter | >50* | >50* | >50* | 15* | 5* | 3* |
| 8. (6) followed by smoothing | >50* | >50* | >50* | 10* | 3* | 2* |
| 9. (8) followed by Viterbi Decoder | >50* | >50* | >50* | 4* | 2* | 1* |
| 10. Pre-detection Kalman Filter | 20* | 10 | 5* | 0* | 0* | 0* |
| 11. (10) followed by Viterbi Decoder | 15* | 4* | 2* | 0* | 0* | 0* |
| 12. Typical Operator | ? | ? | ? | ? | 10* | 5* |
| 13. Good Operator | ? | ? | 10 | 5 | 1 | 0* |

Letter error rates (%) *Projected

obtained by applying the respective improvement ratios actually obtained for non-coherent demodulation. In the case of the pre-detection Kalman filter (row 10), a 10% error rate was actually obtained for the -15 dB entry, and no errors were obtained for the -13 dB entry. However, since the small sample size tends to decrease the confidence in these figures, these rates were increased on a subjective basis by observing the quality of the output signal. Again the subsequent rates for Viterbi decoding (row 11) were obtained by applying the previously determined improvement ratio. The figures for the human operators are the author's estimate based on the error rates obtained previously for the amateur radio operators. A degradation factor of 3-6 dB should probably be added to the SNR's shown for typical field-operation performance.

In summary it may be concluded that the performance of a good operator can be approached by a processing scheme employing pre-detection 100 Hz bandpass filtering followed by discrete Kalman filtering and a Viterbi decoder. The performance of a typical operator can be obtained by either the 230-D or by coherent demodulation preceded by 100 Hz bandpass filtering, with further improvement provided by Kalman filtering, smoothing and Viterbi decoding.

## B. RECOMMENDATIONS

A more appropriate model for the demodulated baseband signal would be one which incorporates the exponential

rise-times and fall-times of the signal pulses instead of the abrupt rise-times and fall-times as used here. A filter incorporating such a model would probably have the effect of reducing the errors caused by extremely short dot insertions during letter-spaces and word-spaces. Better estimators for received demodulated signal noise power and for character duration mean values and/or probability densities should be tested, although it is felt that little advantage would be gained from more sophisticated techniques in an actual operational environment. A third (or higher) order Markov model of the code would undoubtedly show a significant improvement over the first-order model in the effectiveness of the Viterbi decoder at sufficiently low input error rates. Such Markov modeling, along with improved likelihood calculations, deserves further investigation.

The pre-detection Kalman filter shows the greatest advantage, and more investigation of such filtering, and possibly smoothing, with the goal of making the processor real-time, should be undertaken. Down-conversion of the IF signal to 100 Hz or so followed by a 100 Hz bandpass filter, and sampling at 200-400 samples/sec, may be the solution.

# APPENDIX A

## THE DISCRETE KALMAN FILTER

The discrete Kalman filter is an extension of the more familiar Wiener, minimum mean-square-error, matched filter to nonconstant coefficient multivariable systems with nonstationary noise, implemented in sequential, or recursive, form. The result of every processing cycle is the current estimate of the state under consideration. As each new observation is made, the current estimate is updated to reflect the information content of this new measurement. A brief outline of the derivation of the filter algorithm is presented below.

The message model is described by the linear vector difference equation:

$$\underline{x}(k+1) = \underline{\phi}(k)\underline{x}(k) + \underline{\Gamma}(k)\underline{w}(k)$$

where the input noise, or random forcing function, $\underline{w}$ is a zero-mean white-noise process, with covariance

$$\text{cov}[\underline{w}(k),\underline{w}(j)] = \underline{Q}(k)\delta_k(k-j)$$

The observation, or measurement, model is given by the linear algebraic equation

$$\underline{z}(k) = \underline{H}(k)\underline{x}(k) + \underline{v}(k)$$

where $\underline{v}(k)$ is also a zero-mean white-noise process, with

$$\text{cov}[\underline{v}(k),\underline{v}(j)] = \underline{R}(k)\delta_k(k-j) \quad .$$

For simplicity, it is assumed that $\underline{w}$ and $\underline{v}$ are uncorrelated, and that $\underline{w}$ and the initial value of $\underline{x}$ are uncorrelated.

The first step in obtaining the estimate at time k is to predict ahead from the estimate obtained at time k-1. This prediction may be expressed as the conditional expectation:

$$\hat{\underline{x}}(k|k-1) = E[\underline{x}(k)|\underline{z}(k-1)] \quad .$$

But since the measurement $\underline{z}(k-1)$ is embedded in the previous estimate, $\underline{x}(k-1|k-1)$, this prediction may be expressed as

$$\underline{x}(k|k-1) = \underline{\phi}\hat{\underline{x}}(k-1|k-1) + \underline{\Gamma}E[\underline{w}(k-1)] \quad .$$

Since $E[\underline{w}(k)] = 0$, this reduces to

$$\hat{\underline{x}}(k|k-1) = \underline{\phi}\hat{\underline{x}}(k-1|k-1) \quad .$$

The estimate at time k may then be determined by considering it to be a summation of this prediction plus a correction term employing the measured value:

$$\hat{\underline{x}}(k|k) = \underline{\phi x}(k-1|k-1) + \underline{G}(k)[\underline{z}(k) - \hat{\underline{z}}(k|k-1)] \qquad \text{(A-1)}$$

where $\hat{\underline{z}}(k|k-1)$ is determined as follows:

$$\hat{\underline{z}}(k|k-1) = E[\underline{z}(k)|\underline{z}(k-1)]$$

$$= E[(\underline{Hx}(k) + \underline{v}(k))|\underline{z}(k-1)]$$

$$= E[\underline{Hx}(k)|\underline{z}(k-1)] + E[\underline{v}(k)|\underline{z}(k-1)] \quad .$$

Under the given assumptions, this reduces to:

$$\hat{\underline{z}}(k|k-1) = \underline{H}\ E[\underline{x}(k)|\underline{z}(k-1)]$$

$$= \underline{H}\ \underline{\phi}\ \underline{x}(k-1|k-1) \quad .$$

The filter equation (A-1) then becomes:

$$\hat{\underline{x}}(k|k) = \underline{\phi}\hat{\underline{x}}(k-1|k-1) + \underline{G}(k)[\underline{z}(k) - \underline{H\phi}\hat{\underline{x}}(k-1|k-1)]$$

where $\underline{G}(k)$ for optimum filtering is yet to be determined.

It is desired to determine $\underline{G}(k)$ such that the resulting estimates are optimum in a minimum mean-square-error sense. Since the error in the estimate is given by:

$$\underline{x}(k) - \hat{\underline{x}}(k|k) \ ,$$

the covariance matrix of estimation error is

$$\underline{V}(k|k) \triangleq E[(\underline{x}(k) - \hat{\underline{x}}(k|k))(\underline{x}(k) - \hat{\underline{x}}(k|k))^T] \quad .$$

Minimizing a scalar quadratic form based on this matrix will yield the optimum gain $\underline{G}(k)$. Derivations are available in several texts [4], [11], and will not be presented here. The optimum gain results in the following algorithm for the discrete Kalman filter.

Gain:

$$\underline{G}(k) = \underline{V}(k|k-1)\ \underline{H}^T\ [\underline{H}\ \underline{V}(k|k-1)\underline{H}^T + \underline{R}]^{-1}$$

Estimation:

$$\hat{\underline{x}}(k|k) = \hat{\underline{x}}(k|k-1) + \underline{G}(k)[\underline{z}(k) - \underline{H}\hat{\underline{x}}(k|k-1)]$$

Estimation variance:

$$\underline{V}(k|k) = [\underline{I} - \underline{G}(k)\underline{H}]\ \underline{V}(k|k-1)$$

Prediction variance:

$$\underline{V}(k+1|k) = \underline{\phi}\ \underline{V}(k|k)\ \underline{\phi}^T + \underline{Q}(k)$$

Prediction:

$$\hat{\underline{x}}(k+1|k) = \underline{\phi}\ \hat{\underline{x}}(k|k) \quad .$$

## APPENDIX B

## THE VITERBI ALGORITHM

An instructive and informative description of the Viterbi algorithm is presented in [8]; the following outline is based on this article. The MAP sequence estimation problem previously stated in Section VI.A. is formally the same as the problem of finding the shortest path through a certain graph, called a trellis diagram. In this diagram, illustrated in Figure 31, each node corresponds to a distinct state of the Markov signal process and each branch represents a transition to some new state at the next instant of time. In this representation, every possible state sequence corresponds to a unique path through the trellis.

Each branch is assigned the length

$$\lambda(\zeta_k) = -\ln P(x_{k+1}|x_k) - \ln P(z_k|\zeta_k)$$

where

$\zeta_k$ represents a transition from $x_k$ to $x_{k+1}$ .

The $P(x_{k+1}|x_k)$ are the sequence transition probabilities while $P(z_k|\zeta_k)$ are the likelihoods of a particular state.

The algorithm for determining the shortest path through the trellis diagram is a version of forward dynamic programming and is stated as follows:

M = number of states; K = length of input sequence

1. STORAGE:

   $k$ (time index)

   $\underline{\hat{x}}(x_k)$, $1 \leq x_k \leq M$ (survivor terminating in $x_k$)

   $\underline{\Gamma}(x_k)$, $1 \leq x_k \leq M$ (survivor length)

2. INITIALIZATION:

   $k = 0$

   $\underline{\hat{x}}(x_o)=x_o$ , $\underline{\hat{x}}(m)$ arbitrary, $m \neq x_o$

   $\underline{\Gamma}(x_o)=0$; $\underline{\Gamma}(m) = \infty$, $m \neq x_o$

3. RECURSION:

   a. Compute:

   $$\underline{\Gamma}(x_{k+1},x_k) \triangleq \underline{\Gamma}(x_k)+\lambda(\zeta_k), \quad \text{for all } \zeta_k ;$$

   b. Find:

   $$\underline{\Gamma}(x_{k+1}) = \min \underline{\Gamma}(x_{k+1},x_k), \quad \text{for each } x_{k+1};$$

c. Store $\underline{\Gamma}(x_{k+1})$ and the corresponding survivor sequence $\underline{\hat{x}}(x_{k+1})$;

d. Set k to k+1 and repeat until k = K.

As an example [8], the recursive determination of the shortest path through the trellis shown in Figure 32 is shown in Figure 33.

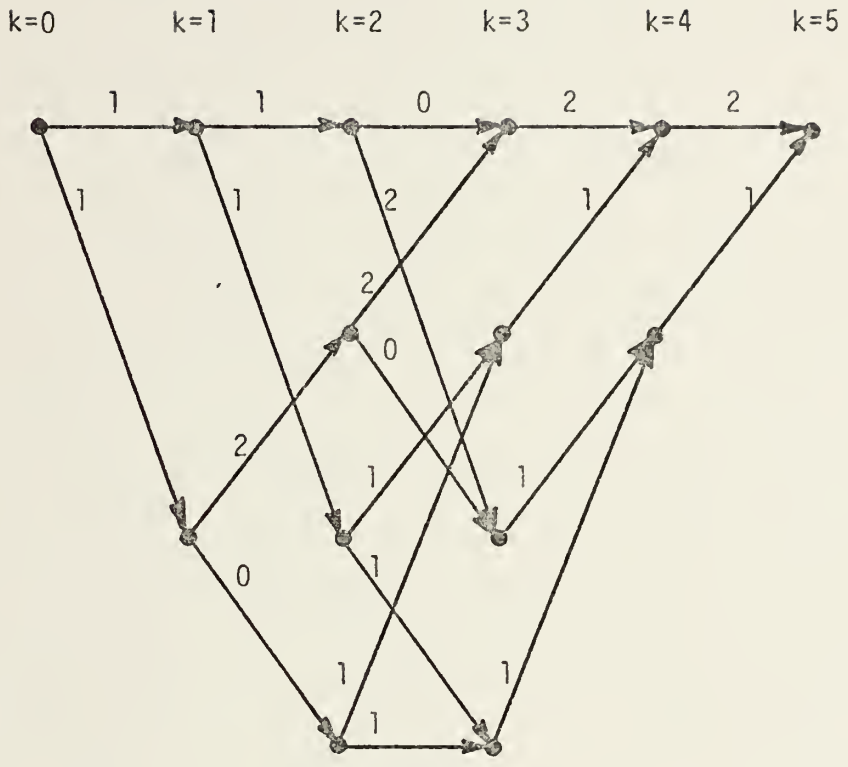



Figure 32. Trellis Diagram with Assigned Lengths

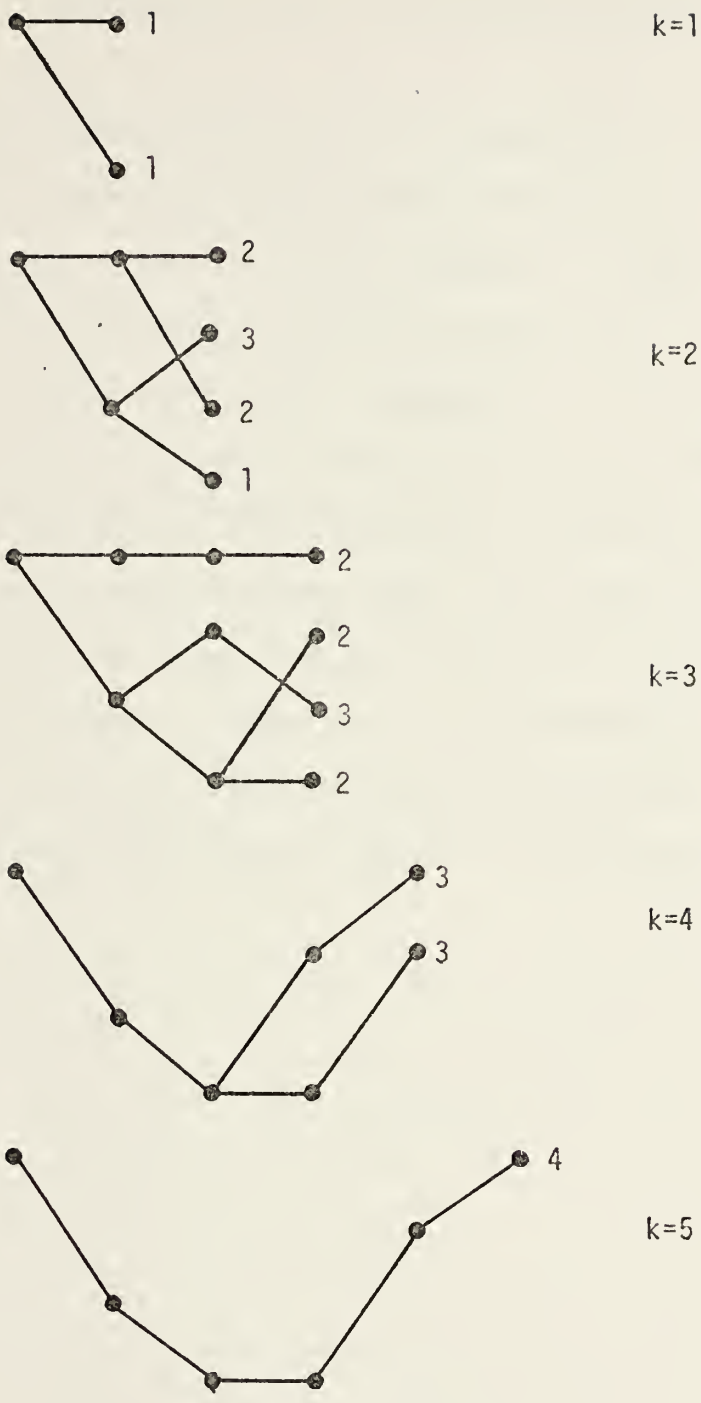



Figure 33. Example of Viterbi Algorithm

# APPENDIX C

## PICKERING 230-D PERFORMANCE EVALUATION

The Pickering 230-D is a currently available automatic decoder and transcriber which includes some front-end processing. The front-end consists of what is essentially a type of coherent detection scheme with a bandwidth of about 180 Hz; the center frequency of the unit tested was at 875 Hz. The letter error rate versus SNR of the 230-D was determined in the laboratory using the model KB-1 keyboard to send perfect code. The error rates were determined for both a 2 kHz bandwidth and a 100 Hz bandwidth and are shown in Table XVI. A performance evaluation of the 230-D for error rates obtained using actual operator-transmitted code with no noise is available in Ref. [2].

TABLE XVI

Pickering 230-D Error Rates

| SNR (in 2 kHz) (dB) | 35 wpm | | 30 wpm | | 25 wpm | |
|---|---|---|---|---|---|---|
| | 2 kHz (%) | 100 Hz (%) | 2 kHz (%) | 100 Hz (%) | 2 kHz (%) | 100 Hz (%) |
| 1.6 | 1 | 3 | 1 | 0 | 0 | 0 |
| 0.4 | 4 | 15 | 4 | 6 | 0 | 0 |
| -0.9 | 29 | 6 | 22 | 13 | 21 | 6 |
| -2.0 | - | 7 | - | 11 | - | 6 |
| -4.4 | - | 7 | - | 7 | - | 4 |
| -6.4 | - | 15 | - | 15 | - | 5 |
| -8.9 | - | 21 | - | 27 | - | 22 |
| -10.5 | - | 60 | - | 78 | - | 61 |
| -12.4 | - | - | - | - | - | - |

Notes:

1) Sample size: Approximately 100 letters in each case

2) Perfect Code

```
$FORTRAN LS,G0
    1: CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    2: C                                                               C
    3: C    THIS PROGRAM IMPLEMENTS THE POST-DETECTION KALMAN          C
    4: C    FILTER AND LINEAR SMOOTHER, ALONG WITH THE REQUIRED        C
    5: C    SIGNAL PARAMETER ESTIMATOR ALGORITHMS.  THE FOLLOWING      C
    6: C    UTILITY SUBROUTINES ARE USED:                              C
    7: C         BUFFERIN - READS DATA ON INPUT TAPE.                  C
    8: C         DAL - D/A CONVERSION AND OUTPUT ROUTINE               C
    9: C    THE FOLLOWING SUBROUTINES ARE PROVIDED:                    C
   10: C         BACK - BACKWARD FILTER ROUTINE                        C
   11: C         STATS- ESTIMATES SIGNAL PARAMETERS                    C
   12: C         VARW - COMPUTES THE VARIANCE OF W(K)                  C
   13: C                                                               C
   14: C    INPUT PARAMETERS AT TIME OF EXECUTION ARE :                C
   15: C         INITIAL ESTIMATE OF T                                 C
   16: C         THE PARAMETERS T1 AND T2                              C
   17: C         THE DOT-DURATION DENSITY PARAMETER, LDOT              C
   18: C         THE DASH-DURATION DENSITY PARAMETER, DDASH            C
   19: C                                                               C
   20: CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
   21: C
   22:       REAL LDOT,MEANX,MEAN,MEANXB,MNX1,MNX2,MERIT
   23:       DIMENSION XHSMN1(50),XHSM(150),XHSMN2(100),XHSMD(100)
   24:       DIMENSION EVARF2(250),XHATF2(250),EVARF(250)
   25:       DIMENSION XB(125),PVARB2(125),XHATB2(125)
   26:       DIMENSION IBUF(500),XHATF(250)
   27:       DIMENSION XHATB1(250),PVARB1(250),XB(250),XHT(10)
   28:       DIMENSION MERIT(4,2,12),TRANS(11)
   29:       COMMON/BLOCK1/XNT,XNTC,XHSM,XHSMD,IJ,IJ1,IJ2,IJL,IJK,
   30:      1TS,TDS,FMEANS,NSHORT
   31:       COMMON/BLOCK2/NELEM,MERIT
   32:       COMMON/BLOCK3/TRANS
   33: C
   34:       NAMELIST LDOT,DDASH,T,T1,T2
   35: C    ENTER TRANSITION PROBABILITIES FOR VITERBI DECODER
   36: C
   37:       TRANS(1)=-ALOG(0.682)
   38:       TRANS(2)=-ALOG(0.318)
   39:       TRANS(3)=-ALOG(0.504)
   40:       TRANS(4)=-ALOG(0.396)
   41:       TRANS(5)=-ALOG(0.25)
   42:       TRANS(6)=-ALOG(0.25)
   43:       TRANS(7)=-ALOG(0.25)
   44:       TRANS(8)=-ALOG(0.25)
   45:       TRANS(9)=-ALOG(0.684)
   46:       TRANS(10)=-ALOG(0.316)
   47:       TRANS(11)=-ALOG(0.1)
   48:    39 CONTINUE
   49: C
   50: C    INPUT PARAMETERS
   51: C
```

```
52:       OUTPUT(101) 'ENTER T, T1, AND T2  (MSEC)'
53:       OUTPUT(101) 'ENTER LOOT AND DDASH (MSEC)'
54:       INPUT(101)
55: C
56: C   INITIALIZE VARIABLES AND INDICES
57: C
58:       T=T/2. ; LOOT=LOOT/2. ; DDASH=DDASH/2.
59:       T1=T1/2. ; T2=T2/2.
60:       A=LOOT/T
61:       B=DDASH/(3.*T)
62:       XOUTL=0. ; MEANXB=0.5 ; MEANX=0.5 ; XT=T ; IT=T
63:       XTD=3.*T
64:       IX=0
65:       KS1=KS2=0
66:       VA1=VA2=1.
67:       S1=S2=.5
68:       SS1=SS2=.4
69:       MNX1=MNX2=0.5
70:       IM1=IM2=IS1=IS2=0
71:       JT=JT2=JTD=JTD2=10
72:       JTC2=JTC=10
73:       NELEM=NSHORT=0
74:       IJ=IJ1=IJ2=II=III=II2=0
75:       SMN1=SMN2=0.
76:       XHSM1=XHSM2=0.
77:       XHSMN1(1)=0. ; XHSMN2(1)=0.
78:       IXN=0
79:       PVAR=EVAR=VARX=0.5
80:       KIND=0 ; KINDF=0 ; JIND=0 ; JINDF=0
81:       Q=1. ; XHATP=0. ; IXHL=0
82:       FMIN=.05 ; XT1=XT2=T
83:       XTD1=XTD2=3.*T
84:       XTC1=XTC2=T
85:       XTA=XTC=T
86:       DO 2 K1=1,4
87:       DO 2 K2=1,2
88:       DO 2 K3=1,12
89:  2    MERIT(K1,K2,K3)=0.
90:  38   CONTINUE
91: C
92: C   INPUT DATA FROM TAPE
93: C
94:  112  CALL BUFFERIN(1,1,IBUF,500,IERR)
95:    1  IF (IERR.EQ.1) GO TO 1
96:       GO TO (1,111,310,112), IERR
97:  111  DO 999 INDE=1,2
98:       INDE1=(INDE-1)*250+1
99:       INDE2=INDE*250
100: C
101: C   BEGIN PROCESSING
102: C
103:       DO 999 INDEX=INDE1,INDE2
104:       X1=FLOAT(IBUF(INDEX))/2**23
```

```
105: C
106: C   UNPROCESSED OUTPUT, XOUT1
107: C
108:       XOUT1=0.
109:       IF(X1.GE.MEANX) XOUT1=1.
110: C
111: C   IX IS THE RUNNING TIME INDEX
112: C
113:       IX=IX+1
114:       IF(IX.GT.250) IX=1
115: C
116: C   IXN INHIBITS SMOOTHING ALGORITHM UNTIL SUFFICIENT
117: C   DATA IS OBTAINED (500 SAMPLES)
118: C
119:       IXN=IXN+1
120:       IF(IXN.GT.500) IXN=501
121: C
122: C   STORE INPUT SAMPLES IN REVERSE ORDER FOR BACKWARD
123: C   FILTERING
124: C
125:       IF(IX.GT.125) XB1=XBB(IX-125) ; GO TO 9
126:       XB1=XB(IX)
127:       XBB(126-IX)=XB(251-IX)
128:  9    XB(251-IX)=X1
129:       CALL STATS
130:       VARXF=VARX
131: C
132: C   FORWARD FILTER ALGORITM
133: C
134:  399  GAINF=PVAR/(PVAR+VARXF)
135:       EVAR=(1.-GAINF)*PVAR
136:       XHAT=XHATP+GAINF*(X1-MEANX-XHATP)
137: C
138: C   THRESHOLDED FILTERED OUTPUT,XOUT2
139: C
140:       XOUT2=0.
141:       IF(XHAT.GE.0.) XOUT2=1.
142:       IXHAT=-1
143:       IF(XHAT.GE.0.     ) IXHAT=1
144:       KIND=KINDF ; JIND=JINDF ; IXHL=IXHLF
145:       T=XT ; TD=XTD ; LOOT=A*XT ; DDASH=B*XTD
146:       XH=XHAT
147:  350  CALL VARw
148:       QF=Q*(FMEAN**2)
149:       PVAR=EVAR+QF
150:       KINDF=KIND ; JINDF=JIND ; IXHLF=IXHL
151:       XHATP=XP
152: C
153: C   STORE FORWARD-FILTERED ESTIMATES AND VARIANCES FOR
154: C   SMOOTHING
155: C
156:       EVARF1=EVARF2(IX); EVARF2(IX)=EVARF(IX)
157:       EVARF(IX)=EVAR
```

```
158:         XHATF1=XHATF2(IX); XHATF2(IX)=XHATF(IX)
159:         XHATF(IX)=XHAT
160:   351   CALL BACK
161:         CALL VARW
162:         QB=Q*(FMEANB**2)
163:         KINDB=KIND ; JINDB=JIND ; IXHLB=IXHL
164: C
165: C    REVERSE STORED VALUES FROM BACKWARD FILTER FOR
166: C    SMOOTHING
167: C
168:         IF(IX.GT.125) XHTB1=XHATB2(IX-125) ; GO TO 492
169:         XHTB1=XHATB1(IX)
170:         XHATB2(126-IX)=XHATB1(251-IX)
171:   492   CONTINUE
172:         XHATB1(251-IX)=XHATBP
173:         IF(IX.GT.125) PVB1=PVARB2(IX-125) ; GO TO 493
174:         PVB1=PVARB1(IX)
175:         PVARB2(126-IX)=PVARB1(251-IX)
176:   493   CONTINUE
177:         PVARB1(251-IX)=PVARB
178:         XHATBP=XP
179:         PVARB=EVARB+QB
180: C
181: C    SMOOTHING ALGORITHM
182: C
183:         WK=EVARF1/(1.+EVARF1/PVB1)
184:         EVARS=((1.-WK/PVB1)**2)*EVARF1 + (WK**2)/PVB1
185:         WBK=(XHTB1/PVB1)*EVARS
186:         IF(IX.EQ.250) WBK=0.
187:         XHATS=XHATF1/(1.+EVARF1/PVB1) +WBK
188:         IF(IXN.LE.500) XHATS=0. ; TS=XT ; TDS=XTD
189: C
190: C    AVERAGE AND STORE SMOOTHED ESTIMATE AT EACH POINT FOR
191: C    USE LATER IN LIKELIHOOD COMPUTATIONS
192: C
193:         IIT=0.75*TS
194:         II1=II1+1
195:         IF(II1.GT.150) II1=1
196:         II=II+1
197:         IF(II.GT.50) II=1
198:         IF(II.GT.IIT) II=1
199:         XHS1=XHSMN1(II)
200:         XHSMN1(II)=XHATS
201:         XHSM1=XHSM1+(XHATS-XHS1)/(0.75*TS)
202:         XHSM(II1)=XHSM1
203:         IIT2=0.67*TDS
204:         II2=II2+1
205:         IF(II2.GT.100) II2=1
206:         IF(II2.GT.IIT2) II2=1
207:         II3=II3+1
208:         IF(II3.GT.100) II3=1
209:         XHS2=XHSMN2(II2)
210:         XHSMN2(II2)=XHATS
```

```
211:        XHSM2=XHSM2+(XHATS-XHS2)/(0.67*TDS)
212:        XHSMD(II3)=XHSM2
213:        IJ=II1-10
214:        IF(IJ.LE.0) IJ=150+IJ
215:        IJ1=II1-5-IIT
216:        IF(IJ1.LE.0) IJ1=150+IJ1
217:        IJ2=II3-5
218:        IF(IJ2.LE.0) IJ2=100+IJ2
219:        XSUM=0.
220:        DO 920 I=2,10
221:        XSUM=XSUM+XHT(I)
222:  920   XHT(I-1)=XHT(I)
223:        XSUM=XSUM+XHATS
224:        XHT(10)=XHATS
225:        XSUM=XSUM/10.
226:        XOUT=0.
227:        IF(XSUM.GT.0.0000001) XOUT=1.
228: C
229: C   ESTIMATE T AND TD
230: C
231: C   XNT IS THE MARK DURATION COUNT
232: C   XNTC IS THE SPACE DURATION COUNT
233: C
234:        KND=0
235:        IF((XOUT-XOUTL).LE.-0.99) KND=1 ; GO TO 900
236:        IF(XOUT.GE. 0.99) NT=NT+1 ; GO TO 990
237:        NT=0
238:        GO TO 990
239:  900   XNT=NT
240:        NT=0
241:        IF((XNT.LT.0.5*T1).OR.(XNT.GT.6.*T2)) GO TO 990
242:        IF(XNT.LE.(3.*T1 + T2)/2.) GO TO 991
243:        JTD=JTD+1
244:        IF(JTD.EQ.10)JTD2=0 ; XTD2=XTD1
245:        IF(JTD.EQ.20) JTD=1
246:        XTD1=XTD1+(XNT-XTD1)/JTD
247:        JTD2=JTD2+1
248:        IF(JTD2.EQ.10) JTD1=0 ; XTD1=XTD2
249:        IF(JTD2.EQ.20) JTD2=1
250:        XTD2=XTD2+(XNT-XTD2)/JTD2
251:        XTD=XTD1
252:        IF(JTD2.GT.JTD1) XTD=XTD2
253:        IF(XTD.LT.(2.*XT)) XTD=2.*XT; XTD1=XTD2=XTD;
254:       1JTD=JTD2=5
255:        GO TO 990
256:  991   JT=JT+1
257:        IF(JT.EQ.10) JT2=0 ; XT2=XT1
258:        IF(JT.EQ.20) JT=1
259:        XT1=XT1+(XNT-XT1)/JT
260:        JT2=JT2+1
261:        IF(JT2.EQ.10) JT=0 ; XT1=XT2
262:        IF(JT2.EQ.20) JT2=1
263:        XT2=XT2+(XNT-XT2)/JT2
```

```
264:        XTA=XT1
265:        IF(JT2.GT.JT) XTA=XT2
266:   990  CONTINUE
267:        IF((XOUT-XOUTL).GE.0.99) KND=1 ; GO TO 1000
268:        IF(XOUT.LE.0.01) NTC=NTC+1 ; GO TO 1090
269:        NTC=0 ; GO TO 1090
270:   1000 XNTC=NTC
271:        NTC=0
272:        IF((XNTC.LT.0.5*T1).OR.(XNTC.GT.6.*T2)) GO TO 1090
273:        IF(XNTC.LE.(3.*T1+T2)/2.) JTC=JTC+1; GO TO 1091
274:        GO TO 1090
275:   1091 IF(JTC.EQ.10) JTC2=0 ; XTC2=XTC1
276:        IF(JTC.EQ.20) JTC=1
277:        XTC1=XTC1+(XNTC-XTC1)/JTC
278:        JTC2=JTC2+1
279:        IF(JTC2.EQ.10) JTC=0 ; XTC1=XTC2
280:        IF(JTC2.EQ.20) JTC2=1
281:        XTC2=XTC2+(XNTC-XTC2)/JTC2
282:        XTC=XTC1
283:        IF(JTC2.GT.JTC) XTC=XTC2
284: C
285: C    SENSE SWITCH 1 IS USED TO SELECT VITERBI DECODER
286: C    OPTION DURING PROGRAM EXECUTION
287: C
288:   1090 IF(SENSE SWITCH 1) 1092,1093
289:   1092 IF(NTC.GT.4.*TS) KND=1 ; XNTC=NTC ; NTC=0
290:        IF(KND.EQ.1) CALL LIKELIH00D
291:   1093 CONTINUE
292:        XT=(XTA+XTC)/2.
293:        XOUTL=XOUT
294: C
295: C    OUTPUT VARIABLES TO D/A ROUTINE FOR ANALOG RECORDING
296: C
297:        TNORM=XT/100.
298:        CALL DAL(X1,XOUT1,XOUT2,XHAT,XHTB1,XHATS,XOUT,VARX,
299:       1GF,GAINF,TNORM)
300:   999  CONTINUE
301:        GO TO 38
302:   310  OUTPUT(101) 'END OF RUN ; HIT * TO GO '
303:        INPUT(101)
304:        GO TO 39
305: C
306: C
307: C
308:        SUBROUTINE BACK
309: C
310: C      THIS ROUTINE IS THE BACKWARD FILTER
311: C
312: C    INITIALIZE INDICES FOR BACKWARD FILTER BASED ON
313: C    FINAL VALUES OF FORWARD-FILTERED ESTIMATES
314: C
315:        IF(IX.NE.1) GO TO 490
316:        IF(KINDF.EQ.0) KINDB=0 ; GO TO 360
```

```
317:        KINDB=XT-KINDF
318:        IF(KINDB.LT.0) KINDB=XTD-KINDF
319:        IF(KINDB.LT.0) KINDB=0
320:  360   IF(JINDF.EQ.0) JINDB=0 ; GO TO 361
321:        JINDB=XT-JINDF
322:        IF(JINDB.LT.0) JINDB=XTD-JINDF
323:        IF(JINDB.LT.0) JINDB=0
324:  361   CONTINUE
325:        XHATBP=XHAT ; IXHLB=IXHAT
326:        PVARB=EVAR ; VARXB=VARX ; MEANXB=MEANX ; QB=QF
327:        TS=TB ; TDS=TDB ; FMEANS=FMEANB
328:        FMEANB=FMEAN ; TB=XT ; TDB=XTD
329:  490   CONTINUE
330: C
331: C    FILTER ALGORITHM
332: C
333:        GAINB=PVARB/(PVARB+VARXB)
334:        EVARB=(1.-GAINB)*PVARB
335:        XHATB=XHATBP+GAINB*(XB1-MEANXB-XHATBP)
336:        IXHATB=-1
337:        IF(XHATB.GE.0.     ) IXHATB=1
338:        IXHAT=IXHATB
339:  491   KIND=KINDB ; JIND=JINDB ; IXHL=IXHLB
340:        IXHLL=IXHLLB ; T=TB ; TD=TDB ; LOOT=A*T ; DDASH=B*TDB
341:        XH=XHATB
342:  450   RETURN
343: C
344: C
345: C
346:        SUBROUTINE STATS
347: C
348: C    THIS SUBROUTINE CALCULATES THE INPUT SIGNAL MEAN,
349: C    VARIANCE, AND ESTIMATES THE SIGNAL AMPLITUDE AND SNR
350: C
351: C
352: C    INPUT MEAN AND VARIANCE CALCULATIONS
353: C
354:        IM1=IM1+1
355:        IF(IM1.EQ.500) IM2=0 ; VA2=VA1 ; MNX2=MNX1
356:        IF(IM1.EQ.1000) IM1=1
357:        MNX1=MNX1+(X1-MNX1)/IM1
358:        VA1=VA1+((X1-MEANX)**2-VA1)/IM1
359:        IM2=IM2+1
360:        IF(IM2.EQ.500) IM1=0 ; VA1=VA2 ; MNX1=MNX2
361:        IF(IM2.EQ.1000) IM2=1
362:        MNX2=MNX2+(X1-MNX2)/IM2
363:        VA2=VA2+((X1-MEANX)**2-VA2)/IM2
364:        MEANX=MNX1
365:        VA=VA1
366:        IF(IM2.GT.IM1) MEANX=MNX2 ; VA=VA2
367: C
368: C    SIGNAL AMPLITUDE ESTIMATOR
369: C
```

```
370:         IF(X1.LE.MEANX) GO TO 80
371:         IS1=IS1+1
372:         IF(IS1.EQ.500) IS2=0 ; S2=S1
373:         IF(IS1.EQ.1000) IS1=1
374:         S1=S1+(X1-S1)/IS1
375:         IS2=IS2+1
376:         IF(IS2.EQ.500) IS1=0 ; S1=S2
377:         IF(IS2.EQ.1000) IS2=1
378:         S2=S2+(X1-S2)/IS2
379:         S=S1
380:         IF(IS2.GT.IS1) S=S2
381:         GO TO 81
382:   80    CONTINUE
383:         KS1=KS1+1
384:         IF(KS1.EQ.500) KS2=0 ; SS2=SS1
385:         IF(KS1.EQ.1000) KS1=1
386:         SS1=SS1+(X1-SS1)/KS1
387:         KS2=KS2+1
388:         IF(KS2.EQ.500) KS1=0 ; SS1=SS2
389:         IF(KS2.EQ.1000) KS2=1
390:        .SS2=SS2+(X1-SS2)/KS2
391:         SS=SS1
392:         IF(KS2.GT.KS1) SS=SS2
393:   81    CONTINUE
394:         FMEAN=S-SS
395:         IF(FMEAN.LT.FMIN) FMEAN=FMIN
396: C
397: C    NOISE VARIANCE ESTIMATOR
398: C
399:         VARX=VA-0.9*(FMEAN**2)/4.
400:         IF(VARX.LE.0.) VARX=0.0000001
401:         RETURN
402: C
403: C
404: C
405: C
406:         SUBROUTINE VARW
407: C
408: C    Q ESTIMATION ALGORITHM
409: C    KIND=TIME INDEX, K, FOR MARK PROBABILITIES
410: C    JIND=TIME INDEX, J, FOR SPACE PROBABILITIES
411: C
412: C    (FOR USE WITH PRE-D FILTER, INITIALIZE KIND,JIND TO 50)
413: C
414:         XP=XH; P=1.0
415:         IF(IXHL.EQ.0) GO TO 10
416:         IF(((IXHAT-IXHL).GT.0).AND.(JIND.GT.T-LOOT)) JIND=0;
417:        1KIND=0; GO TO 10
418:         IF((IXHAT-IXHL).GT.0) IXHAT=-IXHAT ; XP=-XH
419:         IF(((IXHAT-IXHL).LT.0).AND.(KIND.GT.T-LOOT)) JIND=0;
420:        1KIND=0; GO TO 10
421:         IF((IXHAT-IXHL).LT.0) IXHAT=-IXHAT ; XP=-XH
422:   10    IF (IXHAT.EQ.1) KIND=KIND+1
```

```
423:       IF(IXHAT.EQ.-1) JIND=JIND+1 ; GO TO 20
424: C
425: C   MARK PROBABILITIES
426: C
427:       XK=KIND
428:       IF(XK.LE.(T-LDOT)) GO TO 11
429:       GO TO 12
430:  11   Q=0.
431:       IF(KIND.EQ.1) XP=FMEAN/2.
432:       GO TO 100
433:  12   IF((XK.LE.(T+LDOT)).AND.(XK.GT.(T-LDOT))) GO TO 13
434:       GO TO 14
435:  13   PWKO1=(T-XK)/(4.*LDOT)+0.75
436:       PWKN1=1. - PWKO1
437:       Q=PWKO1*PWKN1; P=PWKO1; GO TO 100
438:  14   IF((XK.LE.(TD-DDASH)).AND.(XK.GE.(T+LDOT))) Q=0.;
439:      1GO TO 100
440:       IF((XK.LE.(TD+DDASH)).AND.(XK.GE.(TD-DDASH))) GO TO 17
441:       GO TO 18
442:  17   PWKO1=(TD-XK)/(2.*DDASH) + 0.5
443:       PWKN1=1. -PWKO1
444:       Q=PWKO1*PWKN1; P=PWKO1; GO TO 100
445:  18   Q=0.25; GO TO 100
446: C
447: C   SPACE PROBABILITIES
448: C
449:  20   XJ=JIND ; P=0.5
450:       IF(XJ.LE.(T-LDOT)) GO TO 21
451:       GO TO 22
452:  21   Q=0.
453:       IF(JIND.EQ.1) XP=-FMEAN/2.
454:       GO TO 100
455:  22   IF((XJ.LE.(T+LDOT)).AND.(XJ.GE.(T-LDOT))) GO TO 23
456:       GO TO 24
457:  23   PWKOO=(-XJ/(2.*LDOT)+0.5*(1.+T/LDOT))*12./17.+5./17.
458:       PWKNO=1.-PWKOO
459:       Q=PWKOO*PWKNO; P=PWKOO+0.5; GO TO 100
460:  24   IF((XJ.LE.(TD-DDASH)).AND.(XJ.GE.(T+LDOT))) Q=0.;
461:      1GO TO 100
462:       IF((XJ.LE.(TD+DDASH)).AND.(XJ.GE.(TD-DDASH))) GO TO 27
463:       GO TO 28
464:  27   PWKOO=((-XJ/(2.*DDASH)+0.5*(1.+TD/DDASH)))*0.8+0.2
465:       PWKNO=1.-PWKOO
466:       Q=PWKOO*PWKNO; P=PWKOO+0.5; GO TO 100
467:  28   IF(XJ.LE.5.*T) Q=0. ; GO TO 100
468:       PWKOO=25.*T**2/XJ**2
469:       PWKNO=1.-PWKOO; Q=PWKOO*PWKNO; P=PWKOO+0.5
470:  100  CONTINUE
471:       IF(XK.GT.10.*T) KIND=0 ; Q=0.25
472:       IF(XJ.GT.10.*T) JIND=0 ; Q=0.25
473:       IF(P.GT.1.) P=1.0
474:       IXHL=IXHAT
475:       RETURN
```

```
END OF COMPILATION
     1:        SUBROUTINE LIKELIHOOD
     2: C
     3: C    THIS SUBROUTINE CALCULATES THE LIKELIHOODS OF EACH
     4: C    CHARACTER , USING THE METHOD GIVEN IN THE TEXT OF
     5: C    THE THESIS
     6: C
     7: C    MERIT(1,1,N) IS THE DOT LIKELIHOOD
     8: C    MERIT(2,1,N) IS THE ELEMENT-SPACE LIKELIHOOD
     9: C    MERIT(3,1,N) IS THE LETTER-SPACE LIKELIHOOD FOR
    10: C                 COMPUTATIONS MADE ON A CHARACTER VS
    11: C                 CHARACTER BASIS
    12: C    MERIT(3,2,N) IS THE LETTER-SPACE LIKELIHOOD FOR
    13: C                 COMPUTATIONS INVOLVING THREE SHORT
    14: C                 CHARACTERS
    15: C    MERIT(4,1,N) IS THE CHAR VS CHAR DASH LIKELIHOOD
    16: C    MERIT(4,2,N) IS THE THREE SHORT CHAR DASH LIKELIHOOD
    17: C
    18: C
    19:        REAL MERIT
    20:        DIMENSION MERIT(4,2,12),XMERT(3)
    21:        DIMENSION XHSM(150),XHSMD(100),ESTORE(8)
    22:        DIMENSION MARK(11),NSPACE(11)
    23:        COMMON/BLOCK1/XNT,XNTC,XHSM,XHSMD,IJ,IJ1,IJ2,IJL,IJK,
    24:       1TS,TDS,FMEANS,NSHORT
    25:        COMMON/BLOCK2/NELEM,MERIT
    26:        COMMON/BLOCK4/MARK,NSPACE,NCHAR
    27:        NCHAR=NCHAR+1
    28:        MARK(NCHAR)=2.**XNT
    29:        NSPACE(NCHAR)=2.**XNTC
    30: C
    31: C    MAKE TENTATIVE DECISIONS BASED ON THRESHOLDED
    32: C    SMOOTHED OUTPUT
    33: C
    34:        IF(XNT.LT.0.5) GO TO 103
    35:        IF(XNT.GE.2.**TS) EDUR=XNT; NSHORT=0; ESUM=0.;
    36:       1GO TO 102
    37:        IF((XNT.LT.2.**TS).AND.(XNT.GT.0.5)) EDUR=XNT;
    38:       1GO TO 101
    39:  103  IF(XNTC.GE.2.**TS) EDUR=XNTC; NSHORT=0; ESUM=0.;
    40:       1GO TO 102
    41:        IF((XNTC.LT.2.0*TS).AND.(XNTC.GT.0.5)) EDUR=XNTC
    42: C
    43: C    COMPUTATIONS FOR SHORT CHARACTERS
    44: C
    45:  101  NELEM=NELEM+1
    46:        NSHORT=NSHORT+1
    47:        IF(NSHORT.GT.8) NSHORT=0
```

```
      ESTORE(NSHORT)=EDUR
      ESUM=ESUM+EDUR
      IF(NSHORT.GT.3) ESUM=ESUM-ESTORE(NSHORT-3)
      TM1=1.
      IF(EDUR.GT.TS) TM1=2.-(EDUR+TS)/(2.*TS)
      TM2=0.
      IF(EDUR.GT.TDS/3.) TM2=1.-(TDS-EDUR)/(0.67*TDS)
      XMERIT=(XHSM(IJ)/FMEANS+0.5)*TM1*(1.-TM2)
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(1,1,NELEM)=XMERIT
      XMERIT=(1.-(XHSM(IJ)/FMEANS+0.5))*TM1*(1.-TM2)
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(2,1,NELEM)=XMERIT
      XMERIT=(XHSM(IJ)/FMEANS+0.5)*(1.-TM1*(1.-TM2))
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(4,1,NELEM)=XMERIT
      XMERIT=(1.-(XHSM(IJ)/FMEANS+0.5))*(1.-TM1*(1.-TM2))
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(3,1,NELEM)=XMERIT
C
C   MODIFIED LIKELIHOOD COMPUTATIONS FOR 3 OR MORE
C   SHORT CHARACTERS IN SUCCESSION
C
      IF(NSHORT.LT.3) GO TO 300
      IF(NELEM.LT.3) GO TO 300
      DO 200 I=1,3
  200 XMERT(I)=AMAX(MERIT(1,1,NELEM-3+I),
     1MERIT(2,1,NELEM-3+I))
      XMRT=(1.-XMERT(1))*(1.-XMERT(2))*(1.-XMERT(3))
      XMRT=XMRT**0.33333
      TM2=1.
      IF(ESUM.LT.TDS) TM2=1.-(TDS-ESUM)/(0.67*TDS)
      IF(TM2.LT.0.) TM2=0.
      XMERIT=(XHSMD(IJ2)/FMEANS+0.5)*TM2*XMRT
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(4,2,NELEM-2)=XMERIT
      XMERIT=(1.-(XHSMD(IJ2)/FMEANS+0.5))*TM2*XMRT
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(3,2,NELEM-2)=XMERIT
      GO TO 300
C
C   COMPUTATIONS FOR LONG CHARACTERS
C
  102 NELEM=NELEM+3
      IJK=IJ-EDUR/3.
      IF(IJK.LE.0) IJK=150+IJK
      IJL=IJ-(2.*EDUR)/3.
```

```
      IF(IJL.LE.0) IJL=150+IJL
      TM1=0.
      IF(EDUR.LT.3.*TS) TM1=2.-(EDUR+TS)/(2.*TS)
      IF(TM1.LT.0.) TM1=0. ; IF(TM1.GT.1.) TM1=1.
      TM2=1.
      IF(EDUR.LT.TDS) TM2=1.-(TDS-EDUR)/(0.67*TDS)
      XMERIT=(XHSMD(IJ2)/FMEANS+0.5)*TM2*(1.-TM1)
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(4,2,NELEM-2)=XMERIT
      XMERIT=(1.-(XHSMD(IJ2)/FMEANS+0.5))*TM2*(1.-TM1)
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(3,2,NELEM-2)=XMERIT
C
C   MODIFIED COMPUTATIONS FOR DOT AND ELEMENT-SPACE
C   LIKELIHOODS
C
      IF(EDUR/3..LT.2.5*TS) GO TO 300
      XMERIT=(XHSM(IJL)/FMEANS+0.5)*(1.-TM2*(1.-TM1))
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(1,1,NELEM-2)=XMERIT
      XMERIT=(1.-(XHSM(IJL)/FMEANS+0.5))*(1.-TM2*(1.-TM1))
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(2,1,NELEM-2)=XMERIT
      XMERIT=(XHSM(IJK)/FMEANS+0.5)*(1.-TM2*(1.-TM1))
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(1,1,NELEM-1)=XMERIT
      XMERIT=(1.-(XHSM(IJK)/FMEANS+0.5))*(1.-TM2*(1.-TM1))
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(2,1,NELEM-1)=XMERIT
      XMERIT=(XHSM(IJ)/FMEANS+0.5)*(1.-TM2*(1.-TM1))
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(1,1,NELEM)=XMERIT
      XMERIT=(1.-(XHSM(IJ)/FMEANS+0.5))*(1.-TM2*(1.-TM1))
      IF(XMERIT.LT.0.) XMERIT=0.
      IF(XMERIT.GT.1.) XMERIT=1.
      MERIT(2,1,NELEM)=XMERIT
 300  XNTC=XNT=0.
      IF(NELEM.GE.9) GO TO 301
      RETURN
C
C   TAKE THE LOGARITMS FOR USE BY THE DECODER
C   THE NUMBER 200 IS USED FOR INFINITY
C
 301  DO 501 I1=1,4
      DO 501 I2=1,NELEM
      DO 501 I3=1,2
```

```
154:        IF(MERIT(I1,I3,I2).LE.0.) MERIT(I1,I3,I2)=200.;
155:       1G0 T0 501
156:        MERIT(I1,I3,I2)=-ALOG(MERIT(I1,I3,I2))
157:  501   CONTINUE
158:        CALL DECODER
159: C
160: C    INITIALIZE LIKELIHOODS FOR NEXT ITERATION
161: C
162:        NCHAR=0
163:        DO 600 I=1,11
164:        MARK(I)=NSPACE(I)=0
165:  600   CONTINUE
166:        DO 505 I1=1,4
167:        DO 505 I2=1,12
168:        DO 505 I3=1,2
169:  505   MERIT(I1,I3,I2)=0.
170:        NELEM=0
171:        RETURN
172:        END
```

```
END OF COMPILATION
     1:       SUBROUTINE DECODER
     2: C
     3: C   THIS SUBROUTINE IMPLEMENTS THE
     4: C   VITERBI ALGORITHM
     5: C
     6: C
     7:       REAL LSAV,MERIT,LAMDA,LENGTH,NU
     8:       REAL LNGTH1,LNGTH2,LNGTH3,LNGTH4,LNGTH5,LNGTH6
     9:       INTEGER SEQSAV,CHAR
    10:       DIMENSION LSAV(6,12),ISAVE(6,12),WSAVE(5)
    11:       DIMENSION TRANS(11),XTRANS(11)
    12:       DIMENSION LENGTH(6),YLNGTH(6),WLNGTH(6),ZLNGTH(6)
    13:       DIMENSION MERIT(4,2,12),CHAR(6,12)
    14:       DIMENSION MARK(11),NSPACE(11)
    15:       DIMENSION NSEQ(12),NSQ(12),SEQSAV(5,12)
    16:       COMMON/BLOCK2/NELEM,MERIT
    17:       COMMON/BLOCK3/TRANS
    18:       COMMON/BLOCK4/MARK,NSPACE,NCHAR
    19:       DATA L/0/
    20: C
    21: C        INITIALIZATION
    22: C
    23:       DO 80 I=1,12
    24:  80   NSEQ(I)=NSQ(I)=0
    25: C
    26: C   USE A PRIORI PROBABILITIES OF EACH CHARACTER
    27: C   FOR INITIALIZATION
    28: C
    29:       LENGTH(1)=-ALOG(0.269)
    30:       LENGTH(2)=-ALOG(0.341)
    31:       LENGTH(3)=LENGTH(4)=-ALOG(0.159)
    32:       LENGTH(5)=LENGTH(6)=-ALOG(0.232)
    33:       DO 100 I=1,6
    34:       WLNGTH(I)=ZLNGTH(I)=YLNGTH(I)=200.
    35:  100  CONTINUE
    36:       DO 101 I=1,11
    37:       XTRANS(I)=0.
    38:  101  CONTINUE
    39:       DO 10 I=1,6
    40:       DO 10 J=1,12
    41:       LSAV(I,J)=0.
    42:       ISAVE(I,J)=0
    43:  10   CONTINUE
    44: C
    45: C   SENSE SWITCH 2 IS USED TO SELECT OUTPUT OPTION
    46: C
    47:       IF(SENSE SWITCH 2) 8,9
```

```
 48:  8     WRITE(6,3000)
 49:  3000 FORMAT('1')
 50:       WRITE(6,4010)
 51:  4010 FORMAT(5X,'SURVIVOR SEQUENCES')
 52:       WRITE(6,4000) (I,I=1,11)
 53:  4000 FORMAT(14X,12(I2,5X))
 54:  9    CONTINUE
 55: C
 56: C       MAIN ALGORITHM
 57: C
 58: C    NODE 1 IS A DOT
 59: C    NODE 2 IS AN ELEMENT SPACE
 60: C    NODES 3 AND 4 ARE LETTER-SPACES
 61: C    NODES 5 AND 6 ARE DASHES
 62: C
 63: C
 64:       NELM1=NELEM+1
 65:       DO 900 K=1,NELM1
 66:       IF(K.EQ.2) GO TO 910
 67:       GO TO 911
 68:  910  DO 911 J=1,11
 69:       XTRANS(J)=TRANS(J)
 70:  911  CONTINUE
 71:       IF(K.NE.NELM1) GO TO 920
 72:       DO 920 J=1,11
 73:       XTRANS(J)=0.
 74:  920  CONTINUE
 75: C
 76: C  NODE 1
 77:       LAMDA=LENGTH(2)+XTRANS(3) ; I=2
 78:       NU=LENGTH(3)+XTRANS(5)
 79:       IF(NU.LT.LAMDA) LAMDA=NU ; I=3
 80:       NU=ZLNGTH(4)+XTRANS(5)
 81:       IF(NU.LT.LAMDA) LAMDA=NU ; I=4
 82:       ISAVE(1,K)=I
 83:       LNGTH1=LAMDA+MERIT(1,1,K)
 84:  300  CONTINUE
 85: C
 86: C    NODE 2
 87:       LAMDA=LENGTH(1)+XTRANS(1) ; I=1
 88:       NU=ZLNGTH(4)+XTRANS(6)
 89:       IF(NU.LT.LAMDA) LAMDA=NU ; I=4
 90:       NU=LENGTH(5)+XTRANS(9)
 91:       IF(NU.LT.LAMDA) LAMDA=NU ; I=5
 92:       NU=ZLNGTH(6)+XTRANS(9)
 93:       IF(NU.LT.LAMDA) LAMDA=NU ; I=6
 94:       ISAVE(2,K)=I
 95:       LNGTH2=LAMDA+MERIT(2,1,K)
 96:  301  CONTINUE
 97: C
 98: C    NODE 3
 99:       LAMDA=LENGTH(1)+XTRANS(2) ; I=1
100:       NU=LENGTH(5)+XTRANS(10)
```

```
101:         IF(NU.LT.LAMDA) LAMDA=NU ; I=5
102:         NU=ZLNGTH(6)+XTRANS(10)
103:         IF(NU.LT.LAMDA) LAMDA=NU ; I=6
104:         NU=LENGTH(2)+XTRANS(11)
105:         IF(NU.LT.LAMDA) LAMDA=NU ; I=2
106:         ISAVE(3,K)=I
107:         LNGTH3=LAMDA+MERIT(3,1,K)
108:  302    CONTINUE
109: C
110: C    NODE 4
111:         LAMDA =LENGTH(1)+XTRANS(2) ; I=1
112:         NU=ZLNGTH(4)+XTRANS(7)
113:         IF(NU.LT.LAMDA) LAMDA=NU ; I=4
114:         NU=LENGTH(5)+XTRANS(10)
115:         IF(NU.LT.LAMDA) LAMDA=NU ; I=5
116:         NU=ZLNGTH(6)+XTRANS(10)
117:         IF(NU.LT.LAMDA) LAMDA=NU ; I=6
118:         NU=LENGTH(2)+XTRANS(11)
119:         IF(NU.LT.LAMDA) LAMDA=NU ; I=2
120:         ISAVE(4,K)=I
121:         LNGTH4=LAMDA+MERIT(3,2,K)
122:  303    CONTINUE
123: C
124: C    NODE 5
125:         LAMDA=LENGTH(2)+XTRANS(4) ; I=2
126:         NU=LENGTH(3)+XTRANS(8)
127:         IF(NU.LT.LAMDA) LAMDA=NU ; I=3
128:         NU=ZLNGTH(4)+XTRANS(8)
129:         IF(NU.LT.LAMDA) LAMDA=NU ; I=4
130:         ISAVE(5,K)=I
131:         LNGTH5=LAMDA+MERIT(4,1,K)
132:  304    CONTINUE
133: C
134: C    NODE 6
135:         LAMDA=LENGTH(2)+XTRANS(4) ; I=2
136:         NU=LENGTH(3)+XTRANS(8)
137:         IF(NU.LT.LAMDA) LAMDA=NU ; I=3
138:         NU=ZLNGTH(4)+XTRANS(8)
139:         IF(NU.LT.LAMDA) LAMDA=NU ; I=4
140:         ISAVE(6,K)=I
141:         LNGTH6=LAMDA+MERIT(4,2,K)
142:  305    CONTINUE
143: C
144: C    STORE LENGTHS FOR SURVIVOR SEQUENCES
145: C
146:         LENGTH(1)=LNGTH1
147:         LENGTH(2)=LNGTH2
148:         LENGTH(3)=LNGTH3
149:         LENGTH(4)=LNGTH4
150:         LENGTH(5)=LNGTH5
151:         LENGTH(6)=LNGTH6
152:         DO 11 I=1,6
153:  11     LSAV(I,K)=LENGTH(I)
```

```
154:         DO 200 I=1,6
155:         ZLNGTH(I)=YLNGTH(I)
156:         YLNGTH(I)=WLNGTH(I)
157:         WLNGTH(I)=LENGTH(I)
158:   200   CONTINUE
159:   900   CONTINUE
160:  C
161:  C     DETERMINE MINIMUM LENGTH SEQUENCE FROM THE
162:  C     SIX SURVIVOR SEQUENCES
163:  C
164:         NU=LENGTH(1) ; ISURV=1
165:         DO 901 I=2,6
166:         IF(LENGTH(I).LT.NU) NU=LENGTH(I) ; ISURV=I
167:   901   CONTINUE
168:         WEIGHT=LENGTH(ISURV)
169:         IM=1
170:         M=0
171:         N=ISAVE(ISURV,NELM1)
172:         NSEQ(1)=N
173:   32    CONTINUE
174:         IF((N.EQ.4).OR.(N.EQ.6)) GO TO 30
175:         M=M+1
176:         IF((NELM1-M).LE.0) GO TO 35
177:         NS=ISAVE(N,NELM1-M)
178:         GO TO 31
179:   30    M=M+3
180:         IF((NELM1-M).LE.0) GO TO 35
181:         NS=ISAVE(N,NELM1-M)
182:   31    IM=IM+1
183:         NSEQ(IM)=N=NS
184:         GO TO 32
185:   35    CONTINUE
186:         DO 40 I=1,IM
187:   40    NSQ(I)=NSEQ(IM-I+1)
188:         L=L+1
189:         WSAVE(L)=WEIGHT
190:         DO 71 I=1,IM
191:   71    SEQSAV(L,I)=NSQ(I)
192:         IM1=IM+1
193:         DO 73 I=IM1,12
194:   73    SEQSAV(L,I)=0
195:  C
196:  C     OUTPUT
197:  C
198:         IF(SENSE SWITCH 2) 60,70
199:  C
200:  C     OUTPUT OPTION 1 - DETAILED OUTPUT
201:  C
202:   60    CONTINUE
203:         DO 12 J=1,6
204:   12    WRITE(6,4001) J,(ISAVE(J,K),K=1,NELEM)
205:   4001  FORMAT(8X,12(I1,6X))
206:         WRITE(6,4002)
```

```
4002 FORMAT(/)
     WRITE(6,5000)
5000 FORMAT(5X,'SURVIVOR LENGTHS')
     DO 13 J=1,6
13   WRITE(6,4003) J,(LSAV(J,K),K=1,NELEM)
4003 FORMAT(8X,I1,2X,12(F6.2,1X))
     WRITE(6,4002)
     WRITE(6,1003)
1003 FORMAT(5X,'LIKELIHOODS')
     IJ=1
     JI=0
     DO 903 I=1,2
     JI=JI+1
903  WRITE(6,1002) JI,(MERIT(I,IJ,K),K=1,NELEM)
     DO 905 I=3,4
     DO 905 IJ=1,2
     JI=JI+1
905  WRITE(6,1002) JI,(MERIT(I,IJ,K),K=1,NELEM)
1002 FORMAT(8X,I1,2X,12(F6.2,1X))
     WRITE(6,4002)
     WRITE(6,4021)
4021 FORMAT(63X,'DECODED')
     WRITE(6,4020)
4020 FORMAT(24X,'MARK',3X,'SPACE',12X,'LENGTH',9X,
    1'SEQUENCE')
904  WRITE(6,1000) MARK(1),NSPACE(1),WEIGHT,(NSQ(K),
    1K=2,IM)
1000 FORMAT(20X,I5,3X,I5,11X,F9.3,10X,12I1)
     DO 960 I=2,NCHAR
     WRITE(6,1000) MARK(I),NSPACE(I)
960  CONTINUE
     IF(L.EQ.5) L=0
     GO TO 72
 70  IF(L.NE.5) GO TO 72
C
C   OUTPUT OPTION 2 - DECODED SEQUENCES AND LENGTHS ONLY
C
     L=0
     WRITE(6,6000) (WSAVE(I),I=1,5)
     WRITE(6,6001) ((SEQSAV(I,J),J=2,12),I=1,5)
6000 FORMAT(5X,5(F9.3,2X),/)
6001 FORMAT(5X,5(11I1,1X))
     WRITE(6,6003)
6003 FORMAT(//)
72   CONTINUE
     RETURN
     END
```

```
▵FORTRAN LS,G0
     1: C    THIS PROGRAM IMPLEMENTS THE PRE-DETECTION FILTER
     2: C    DESCRIBED IN THE THESIS TEXT
     3: C
     4:        REAL LD0T,MEANX,IDENT
     5:        DIMENSION IDENT(2,2),GAIN(2),H(2)
     6:        DIMENSION TEMPM(2,2),TEMPM1(2,2),TEMPR(2),TEMPC(2)
     7:        DIMENSION EVAR(2,2),PVAR(2,2),PHI(2,2),PHIT(2,2)
     8:        DIMENSION IBUF(4000)
     9:        NAMELIST LD0T,DDASH,T,T1,T2,FREQ
    10:        OUTPUT(101) 'ENTER T AND FREQ'
    11:        OUTPUT(101) 'LD0T AND DDASH'
    12:        INPUT(101)
    13: C
    14: C    INITIALIZE VARIABLES AND INDICES
    15: C
    16:        XP1=XP2=0.5
    17:        TAU=0.00025
    18:        KT=0
    19:        KIND=JIND=0
    20:        X0UT=0.
    21:        MEANX=0.
    22:        SS=0.
    23:        VARX=0.5
    24:        T=4.*T ; LD0T=4.*LD0T; DDASH=4.*DDASH
    25:        T1=4.*T1 ; T2=4.*T2
    26:        FREQ=6.28319*FREQ
    27:        TD=3.*T
    28:        IXHL=0
    29:        P=1.0
    30:        D0 20 I=1,2
    31:        PHI(I,I)=1.
    32:        PHIT(I,I)=1.
    33:   20   C0NTINUE
    34:        PHI(1,2)=PHIT(2,1)=FREQ*TAU
    35:        PHI(2,1)=PHIT(1,2)=-FREQ*TAU
    36:        D0 21 I=1,2
    37:        D0 21 J=1,2
    38:        IDENT(I,J)=0.
    39:        EVAR(I,J)=0.
    40:        PVAR(I,J)=0.
    41:   21   C0NTINUE
    42:        PVAR(1,1)=PVAR(2,2)=1.
    43:        IDENT(1,1)=IDENT(2,2)=1.
    44:        H(1)=1.
    45:        H(2)=0.
    46:   100  C0NTINUE
    47: C
    48: C    INPUT DATA FROM TAPE
    49: C
    50:   112  CALL BUFFERIN(1,1,IBUF,4000,IERR)
    51:   1    IF(IERR.EQ.1) G0 T9 1
```

```
52:         GO TO (1,111,310,112) IERR
53:  111   DO 999 INDEX=1,4000
54: C
55: C    BEGIN PROCESSING
56: C
57:         X1=FLOAT(IBUF(INDEX))/2**23
58:         X2=X1
59:         X1=P*X1
60:         IF(KT.GT.1000000) KT=0
61:         KT=KT+1
62:         CALL STATS
63: C
64: C    GAIN
65:         CALL MVMULT(PVAR,H,TEMPC)
66:         CALL VVMULT(H,TEMPC,TEMP)
67:         TEMP1=1./(TEMP+VARX)
68:         CALL MVMULT(PVAR,H,TEMPC)
69:         GAIN(1)=TEMPC(1)*TEMP1
70:         GAIN(2)=TEMPC(2)*TEMP1
71: C
72: C    ESTIMATION
73:         XH1=XP1+GAIN(1)*(X1-XP1)
74:         XH2=XP2+GAIN(2)*(X1-XP1)
75: C
76: C    ESTIMATION VARIANCE
77:         CALL VMULT(GAIN,H,TEMPM)
78:         DO 200 I=1,2
79:         DO 200 J=1,2
80:  200    TEMPM(I,J)=IDENT(I,J)-TEMPM(I,J)
81:         CALL MMULT(TEMPM,PVAR,EVAR)
82: C
83: C    PREDICTION VARIANCE
84:         CALL MMULT(EVAR,PHIT,TEMPM)
85:         CALL MMULT(PHI,TEMPM,PVAR)
86: C
87: C    ONE-STEP PREDICTION
88:         XP1=XH1+PHI(1,2)*XH2
89:         XP2=PHI(2,1)*XH1+XH2
90: C
91: C    SQUARE FILTERED ESTIMATE AND LOW-PASS FILTER
92: C
93:         X=XH1**2
94:         XKT=KT
95:         IF(KT.GT.50) GO TO 60
96:         XOUT=XOUT+(1./XKT)*(X-XOUT)
97:         GO TO 61
98:  60     XOUT=XOUT+0.02*(X-XOUT)
99:  61     CONTINUE
100:        IXHAT=-1
101:        IF(XOUT.GE.MEANX) IXHAT=1
102:        CALL VARW
103: C
104: C    OUTPUT VALUES TO D/A ROUTINE FOR ANALOG RECORDING
```

```
105: C
106:       XXHAT=FL0AT(IXHAT)
107:       CALL DAL(X2,X1,XH1,X0UT,GAIN(1),EVAR(1,1),EVAR(2,2),
108:      1VARX,P,XXHAT)
109:  999  C0NTINUE
110:  310  G0 T0 100
111: C
112: C
113: C
114:       SUBR0UTINE STATS
115: C
116: C    THIS SUBR0UTINE ESTIMATES THE N0ISE VARAINCE
117: C
118: C
119:       XKT=KT
120:       IF(KT.GT.4000) G0 T0 70
121:       MEANX=MEANX+(1./XKT)*(X0UT-MEANX)
122:       G0 T0 71
123:  70   MEANX=MEANX+(1./4000.)*(X0UT-MEANX)
124:  71   C0NTINUE
125:       IF(X0UT.GT.MEANX) G0 T0 80
126:       SS=SS+0.01*(X0UT-SS)
127:  80   C0NTINUE
128:       VARX=SS
129:       IF(VARX.LE.0.01) VARX=0.01
130:       RETURN
131:       END
```

```
↑FORTRAN LS,G9
     1:        SUBROUTINE MMULT(A,B,C)
     2: C   MULTIPLY TWO MATRICES
     3:        DIMENSION A(2,2),B(2,2),C(2,2)
     4:        DO 10 I=1,2
     5:        DO 10 J=1,2
     6:   10   C(I,J)=0.
     7:        DO 20 I=1,2
     8:        DO 20 J=1,2
     9:        DO 20 K=1,2
    10:        C(I,J)=A(I,K)*B(K,J)+C(I,J)
    11:   20   CONTINUE
    12:        RETURN
    13:        END
```

```
END OF COMPILATION
      1:        SUBROUTINE MVMULT(A,B,C)
      2: C   MULTIPLY MATRIX BY COLUMN VECTOR
      3:        DIMENSION A(2,2),B(2),C(2)
      4:        DO 10 I=1,2
      5:   10   C(I)=0.
      6:        DO 20 I=1,2
      7:        DO 20 K=1,2
      8:        C(I)=A(I,K)*B(K)+C(I)
      9:   20   CONTINUE
     10:        RETURN
     11:        END
```

```
END OF COMPILATION
      1:        SUBROUTINE VMMULT(A,B,C)
      2: C   MULTIPLY ROW VECTOR BY MATRIX
      3:        DIMENSION A(2),B(2,2),C(2)
      4:        DO 10 I=1,2
      5:   10   C(I)=0.
      6:        DO 20 I=1,2
      7:        DO 20 K=1,2
      8:        C(I)=A(K)*B(K,I)+C(I)
      9:   20   CONTINUE
     10:        RETURN
     11:        END
```

```
END OF COMPILATION
      1:        SUBROUTINE VVMULT(A,B,C)
      2: C   MULTIPLY ROW VECTOR BY COLUMN VECTOR
      3:        DIMENSION A(2),B(2)
      4:        C=0.
      5:        DO 10 I=1,2
      6:        C=A(I)*B(I)+C
      7:   10   CONTINUE
      8:        RETURN
      9:        END
```

```
END OF COMPILATION
      1:        SUBROUTINE VMULT(A,B,C)
      2: C    MULTIPLY COLUMN VECTOR BY ROW VECTOR
      3:        DIMENSION A(2),B(2),C(2,2)
      4:        DO 10 I=1,2
      5:        DO 10 J=1,2
      6:        C(I,J)=A(I)*B(J)
      7:   10   CONTINUE
      8:        RETURN
      9:        END
```

## LIST OF REFERENCES


1. Althoff, W.A., An Automatic Radiotelegraph Translator and Transcriber for Manually Sent Morse, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1973.

2. Deleted.

3. Demetry, J.S., "Notes on the Theory and Applications of Optimal Estimation," unpublished notes, Naval Postgraduate School, Monterey, California, Copyright 1970.

4. Sage, A.P. and Melsa, J.L., Estimation Theory with Applications to Communications and Control, McGraw-Hill, 1971.

5. Fraser, D.C., and Potter, J.E., "The Optimum Linear Smoother as a Combination of Two Optimum Linear Filters," IEEE Trans. Automat. Contr., v. AC-14, no. 4, pp. 387-390, August 1969.

6. Monzingo, R.A., "Discrete Optimal Linear Smoothing for Systems with Uncertain Observations," IEEE Trans. Inform. Theory, v. IT-21, no. 3, pp. 271-275, May 1975.

7. Viterbi, A.J., "Error Bounds for Convolutional Codes and an Asymptotic Optimum Decoding Algorithm," IEEE Trans. Inform. Theory, v. IT-13, no. 2, pp. 260-269, April 1967.

8. Forney, G.D., Jr., "The Viterbi Algorithm," Proc. IEEE, v. 61, no. 3, pp. 268-278, March 1973.

9. Neuhoff, D.L., "The Viterbi Algorithm as an Aid in Text Recognition," IEEE Trans. Inform. Theory, v. IT-21, no. 2, pp. 222-226, March 1975.

10. Gallager, R.G., Information Theory and Reliable Communications, pp. 120-122, Wiley, 1968.

11. Morrison, N., Introduction to Sequential Smoothing and Prediction, McGraw-Hill, 1969.

## INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |
| 2. | Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 3. | Professor C. H. Rothauge, Code 52<br>Department Chairman<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 4. | Associate Professor Stephen Jauregui, Jr.,<br>Code 52Ja<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 10 |
| 5. | Dr. Robert Fossum, Code 023<br>Dean of Research<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 6. | Associate Professor R. Adler, Code 52Ab<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 7. | Professor C. Comstock, Code 53Zk<br>Department of Mathematics<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 8. | Associate Professor J. Ohlson, Code 52Ol<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 9. | Assistant Professor V.M. Powers, Code 52Pw<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |

10. Professor H. Titus, Code 52Ts — 1
Department of Electrical Engineering
Naval Postgraduate School
Monterey, California 93940

11. LT Edison L. Bell — 1
SMC Box 1019
Naval Postgraduate School
Monterey, California 93940

12. Commander, Naval Security Group Command — 1
Naval Security Group Headquarters
3801 Nebraska Ave., N.W.
Washington, D.C. 20390
ATTN: CAPT Osmer, GB

13. Commander, Naval Security Group Command — 5
Naval Security Group Headquarters
3801 Nebraska Ave., N.W.
Washington, D.C. 20390
ATTN: CDR H. Shoemaker, G82
CDR W. Garner
LT W. Althoff
LT J. Malloy
LCDR F. Cleary

14. Commanding Officer — 1
Naval Electronic Systems Security
Engineering Center
Naval Security Station
3801 Nebraska Ave., N.W.
Washington, D.C. 20390
ATTN: Mr. D. Webster

15. Commander, Naval Electronic Systems Command — 2
Naval Electronic Systems Command Headquarters
PME-107
Washington, D.C. 20360
ATTN: CAPT Patterson
CAPT G. Smith

16. Commander, Naval Electronic Systems Command — 5
Naval Electronic Systems Command Headquarters
PME-107-5
Washington, D.C. 20360
ATTN: LCDR R. Todaro
Mr. P. Lowell
Mr. R. LaSage
CWO J. James
CDR A. P. Taylor

17. Commander, Naval Electronic Systems Command — 1
Naval Electronic Systems Command Headquarters
PME-106
Washington, D.C. 20360
ATTN: CAPT J. Pope

18. Commander, Naval Electronics Laboratory Center — 2
San Diego, California 92152
ATTN: Mr. J. Griffin
LT W. Gadino

19. Assistant Commander, Naval Security Group — 1
Ft. George G. Meade, Maryland 20755
ATTN: LT Cumings

20. Assistant Secretary of Defense for Intelligence — 1
The Pentagon
Washington, D.C. 20301
ATTN: Dr. E. O. Brigham

21. Assistant Secretary of Defense for Intelligence — 1
The Pentagon
Washington, D.C. 20301
ATTN: Mr. M. Goulder

22. Director — 2
National Security Agency
Group W
Ft. George G. Meade, Maryland 20755
ATTN: Mr. J. Boone
Dr. J. Cole

23. Director — 4
National Security Agency
Group R
Ft. George G. Meade, Maryland 20755
ATTN: LT T. Reglein
Mr. J. Hull
Mr. D. Howe
Mr. D. Thouin

24. Army Security Agency — 1
Arlington Hall Station
Arlington, Virginia 22212
ATTN: Mr. H. Hovey

25. ASA Field Station — 1
Vint Hill Farms
Warenton, Virginia 22186
ATTN: Dr. White

26. Electromagnetic Systems Laboratories, Inc. 1
495 Java Drive
Sunnyvale, California 94086
ATTN: Mr. W. Phillips

27. Interstate Electronics Corporation 1
707 E. Vermont
Box 3117
Anaheim, California 92803
ATTN: Mr. L. R. Roberts

28. Sanders Associates 2
95 Canal Street
Nashua, New Hampshire 03060
ATTN: Mr. W. Zandi
Mr. T. Huff

29. Sylvania, EDL Systems West 1
P.O. Box 205
Mountain View, California 94040
ATTN: Mr. D. Littler

30. TRW, Inc. 1
2700 Building S
1 Space Park
Redondo Beach, California 90278
ATTN: Dr. Barry Whalen

31. Pickering Radio Company 1
P.O. Box 29
Portsmouth, Rhode Island 02871

32. Naval Communications Station, Rota 1
NAVSECGRU Department
FPO, New York 09540
ATTN: LCDR R. Shields